Home OF Games Studio

[تعلم برمجة الالعاب لمحرك الالعاب Unity3D جزء (٢)]

[تعلم بدون تعقيد – عماد عارف]

[الكتاب مهتم بتعلم اساسيات البرمجة لمحرك الالعاب Unity3D و فهم الدوال و المتغيرات و بعض المفاهيم التي لم يتم شرحها ، وايضا شرح طريقة التعامل مع الأكواد البسيطة لخالق بعض الالعاب]

الكتاب مقدم من : عماد عارف التوي

قناة :

Home OF Games

تعلم البرمجة لمحرك الالعاب Unity صار شبة معدوم في العالم العربي حتى ان اغلب الدروس لم تكن عربية. (Scripting)

للأسف لا توجد دروس عربية لتعلم هذه البرمجة من الاساسيات , الان مدخلاتها ثانية مع العلم انها تستخدم لغة Java Script و #Cالى ان مدخلاتها بخلاف باقي لغات البرمجة.
انا صببت تركيزي على البرمجة فقط , تعلم اساسيات البرنامج بشكل عام شيء سهل ولا يحتاج الى دروس , مايهم الان هو البرمجة لأنها العائق الذي واجة الكثيرين , هذا الكتاب ان شاء الله راح يتقسم الى عدة اجزاء لا استطيع ذكرها الى ان تنتهي الدورة بأذن الله .

أي استفسار عن هذا الكتاب يرجى ارسال رسالة الى هذا البريد الإلكتروني :

emadye11@hotmail.com
or
homeofgamesnews@gmail.com

و ان شاء الله يوصلكم الرد بأسرع وقت.

ان شاء الله تكون هذه الدورة مبسطة بشكل كامل و كل كود راح يكون له شرح خاص, و ان شاء الله قريباً راح يتم رفع دروس بالفيديو الى قناتنا.

ملاحظة : الكتاب الاول كان عبارة عن مقدمة لا أكثر ولا اقل.
بالنسبة لهذا الكتاب ان شاء الله راح يكون البداية في تعلم البرمجة من الاساسيات البسيطة في محرك الالعاب Unity3D 4.5.5

مفهوم Time.deltaTime

مفهوم الـ Time.deltatime معناه : الوقت الحقيقي المستخدم لضرب أي سرعة تعين في البرنامج فيه , هذا بشكل مختصر وللتوضيح .

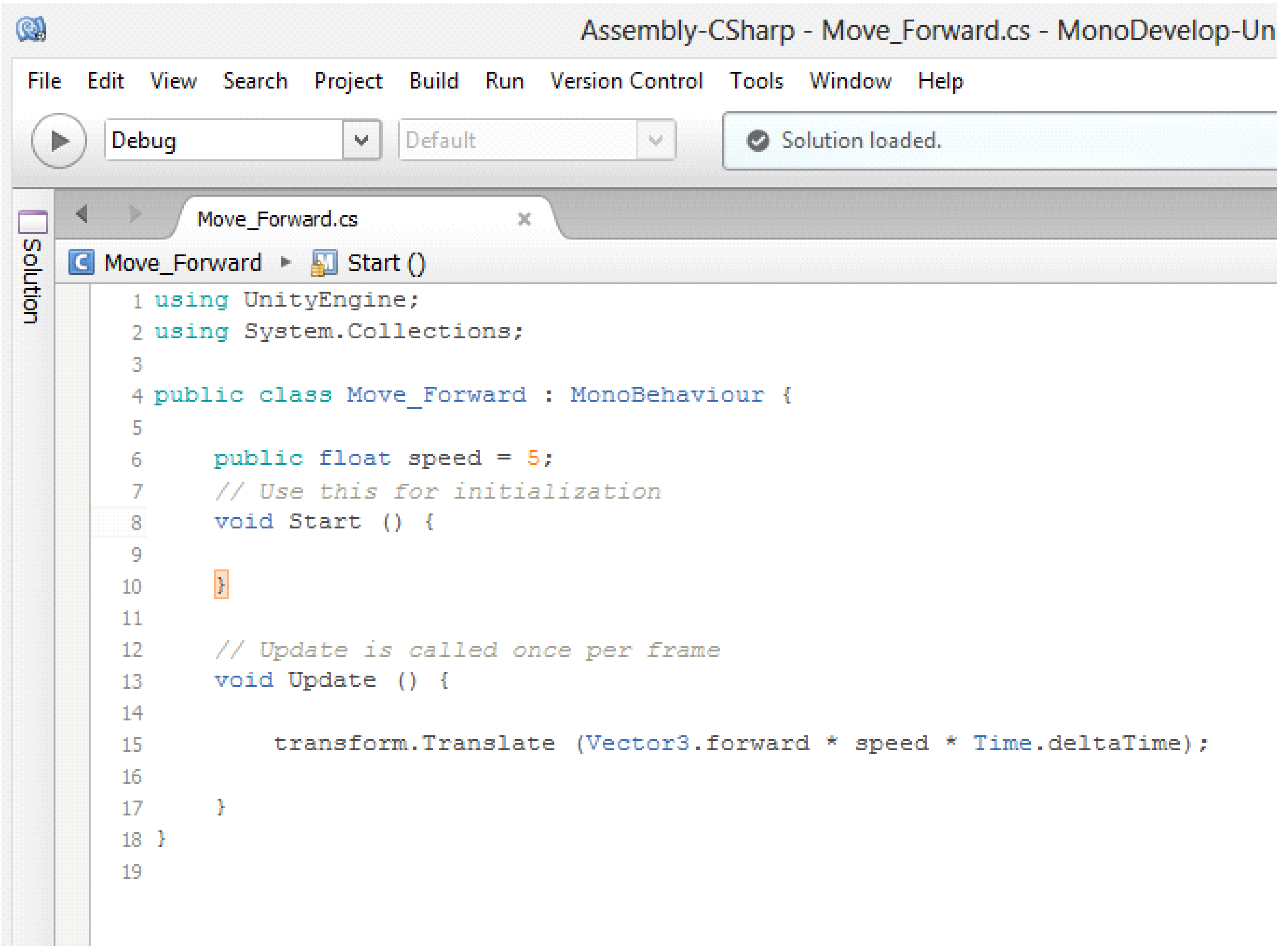
مثلاً : معك متغير من نوع قيمة عددية float يحمل اسم speed و قيمته المعينة = 5 .

الان عندما تقوم بإضافة هذه السرعة في أي ازاحة راح تلاحظ ان السرعة غير اعتيادية أي انها ليست مع اوقت الحقيقي (في كل ثانية)لهذا تأتي ال Time.deltatim لكي تصحح هذه المشكلة و تقوم بضرب اي قيمة فيها و راح تلاحظ ان السرعة صارت في كل ثانية .

مفهومها سهل جرب تعملها في اي برمجة و لاحظ الفرق .

مثال بسيط لكي تقوم بتطبيقه ..

انشئ ملف و قم بتسميته (Move_Forward) او أي اسم تريده , الان قم بإضافة هذا الكود فيه :




```
using UnityEngine;
using System.Collections;

public class Move_Forward : MonoBehaviour {

    public float speed = 5;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

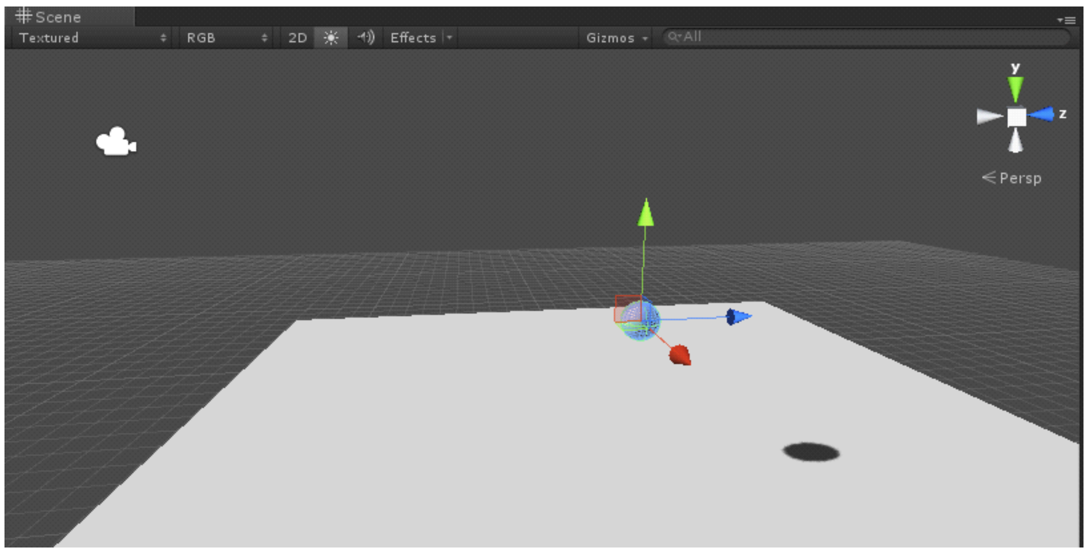
        transform.Translate (Vector3.forward * speed * Time.deltaTime);

    }
}
```

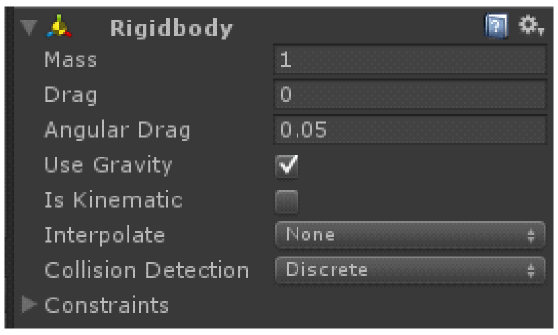
الجاذبية في البرمجة

في درس الحركة السابق عرفنا كيفية تحريك أي جسم باستخدام ال Up و ال forward , في هذا الشرح راح نتعلم كيفية اضافة حركة الى هذا الجسم باستخدام الجاذبية الارضية Rigidbody

مثال : قم بأنشاء مشروع مثل الذي في الصورة التالية :



قم بإضافة خاصية ال Rigidbody في الكرة كما في الشكل:



الان قم بأنشاء ملف برمجي و لاحظ الصورة التالية التي تحتوي على اكواد الشرح .

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class moveRig : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13
14        float moveHorizontal = Input.GetAxis ("Horizontal");
15        float moveVertical = Input.GetAxis ("Vertical");
16        Vector3 movement = new Vector3 (x, y, z);
17        rigidbody.AddForce (movement);
18    }
19 }
20
```

الشرح : تلاحظ في الصورة اننا اضفنا في السطر 14 متغير من نوع float و بإضافة قيمة عددية الى هذه السرعة او نوع الازاحة في النص كما هوا موضح في الصورة , طبعاً ال move Horizontal هو اسم هذا المتغير راح يكون و سيط عند استدعائه لاحقاً , الان بإضافة طريق الادخال في ال input و التي هي (GetAxis) اضافنا الازاحة الافقية المستمرة لعمل حركة مستمرة بشكل سلس , نفس الشيء في السطر 15بتغيير ال Horizontal الى Vertical الان لإضافة هذه الازاحة في المحاور الثلاثة كما في السطر 16 بإضافة ازاحة جديدة باسم movement و التي تحتوي على ازاحة جديدة new Vector3 البعض يتساءل لماذا اضفنا ال new Vector3 هذا لأنه لا يوجد لدينا

ازاحة في البرنامج لنقوم بتعيينها بدلاً عنة , و قمنا بإضافة ال new Vector3 لإضافة ازاحة جديدة في ملف البرمجة طبعا ازاحة من نوع 3 أي 3 محاور الان بفتح ال 3 المحاور تلاحظ ان قيمة الكل هي صفر , (0,0,0) طيب الان عرفنا ان المحور X في البرنامج هو المحور الافقي لهذا نقوم بإضافة ال move Horizontal فيه و المحور Z هو المحور الامامي و الخلفي و نقوم بإضافة الازاحة العمودية فيه , move Vertical في السطر 17 تلاحظ اننا اضافنا الجاذبية Rigidbody و اضافنا قوة الى هذه الجاذبية الان بفتح القوسين يطلب مننا اضافة الجاذبية و القوة الى أي شيئ ؟

طبعاً نحن نريد اضافة الإزاحات التي عيناها في الملف الى هذه القوة, لان بالأساس ازاحة بدون قوة تساوي القيمة 0 طبعا بضرب ال movement في القوة بأي سرعة او يمكنك عمل متغير يحتوي على سرعة معينة , الان لاحظ الصورة التالية فيها كل ما شرحته باختصار :

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class moveRig : MonoBehaviour {
5
6     public float speed = 50;
7     // Use this for initialization
8     void Start () {
9
10    }
11
12    // Update is called once per frame
13    void Update () {
14
15        float moveHorizontal = Input.GetAxis ("Horizontal");
16        float moveVertical = Input.GetAxis ("Vertical");
17        Vector3 movement = new Vector3 (moveHorizontal, 0, moveVertical);
18        rigidbody.AddForce (movement*speed*Time.deltaTime);
19    }
20 }
21
```

ملاحظ : اذا وجدت أي ثقل في الحركة قوم بتغيير قوة الجاذبية او بتنقيصها او بتزيد السرعة في الملف حسب رغبتك

ربط كائن متحرك في كائن

في الكتاب السابق عرفنا طريقة اضافة حركة عادية الى أي كائن

كما في الصورة , للتذكير فقط :

```
using UnityEngine;
using System.Collections;

public class moveRig : MonoBehaviour {

    public float speed = 1;
    // Use this for initialization
    void Start () {

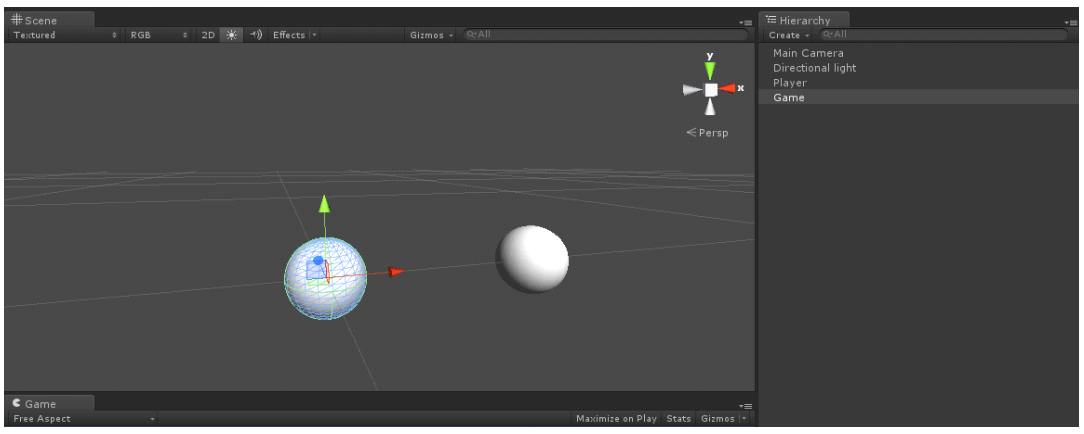
    }

    // Update is called once per frame
    void Update () {

        transform.Translate (new Vector3 (speed,0,0)*Time.deltaTime);
    }
}
```

طيب في الدرس هذا راح نشرح ربط كائن متحرك في كائن آخر كيف نعمل هذا ؟ تابع:

اولاً قم بأنشاء مثل هذا المشروع :



الان قم بأنشاء ملفين سكربت الاول باسم(MovePlayer) او أي اسم تريده فقط للتفريق بين الملفين في الشرح , الان شرح سريع فقط للطريقة التي راح نعمل عليها : اولاً مفهوم كلمة ربط كائن متحرك في كائن , انه يوجد لدينا كائن متحرك في حركة جانبية او افقية او عمودية... الخ, و نقوم بربط هذا الكائن المتحرك في كائن آخر و جعلة يدور حول هذا الكائن , الطريقة المستخدمة بإضافة الـ Transform تابع الشرح .

الان قم بفتح ملف البرمجة و قم بكتاب الأكواد كما هو موضح في الصورة:

```
using UnityEngine;
using System.Collections;

public class MovePlayer : MonoBehaviour {

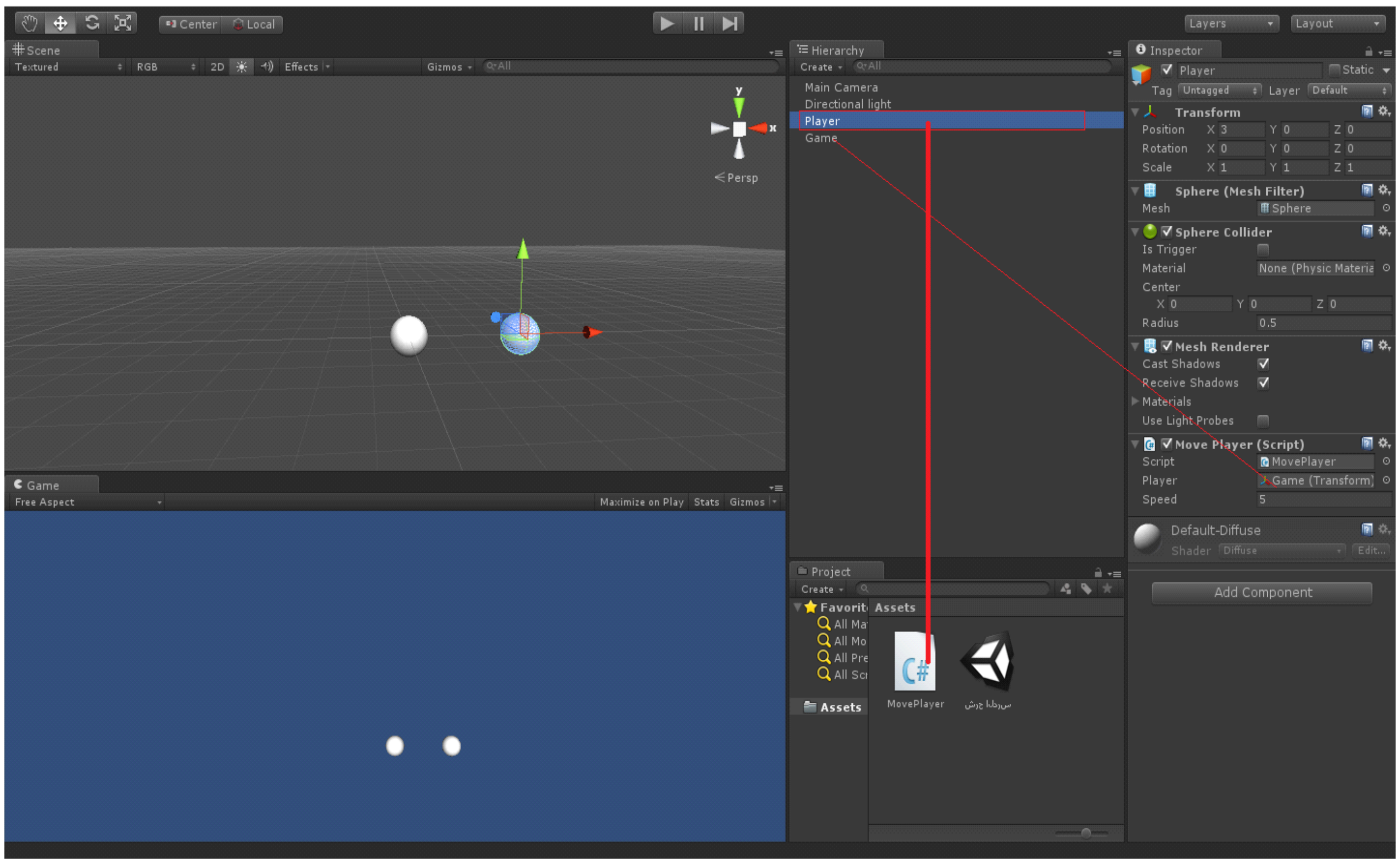
    public Transform Player;
    public float speed = 5;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        transform.Translate (new Vector3 (speed, 0, 0) * Time.deltaTime);
        transform.LookAt (Player);
    }
}
```

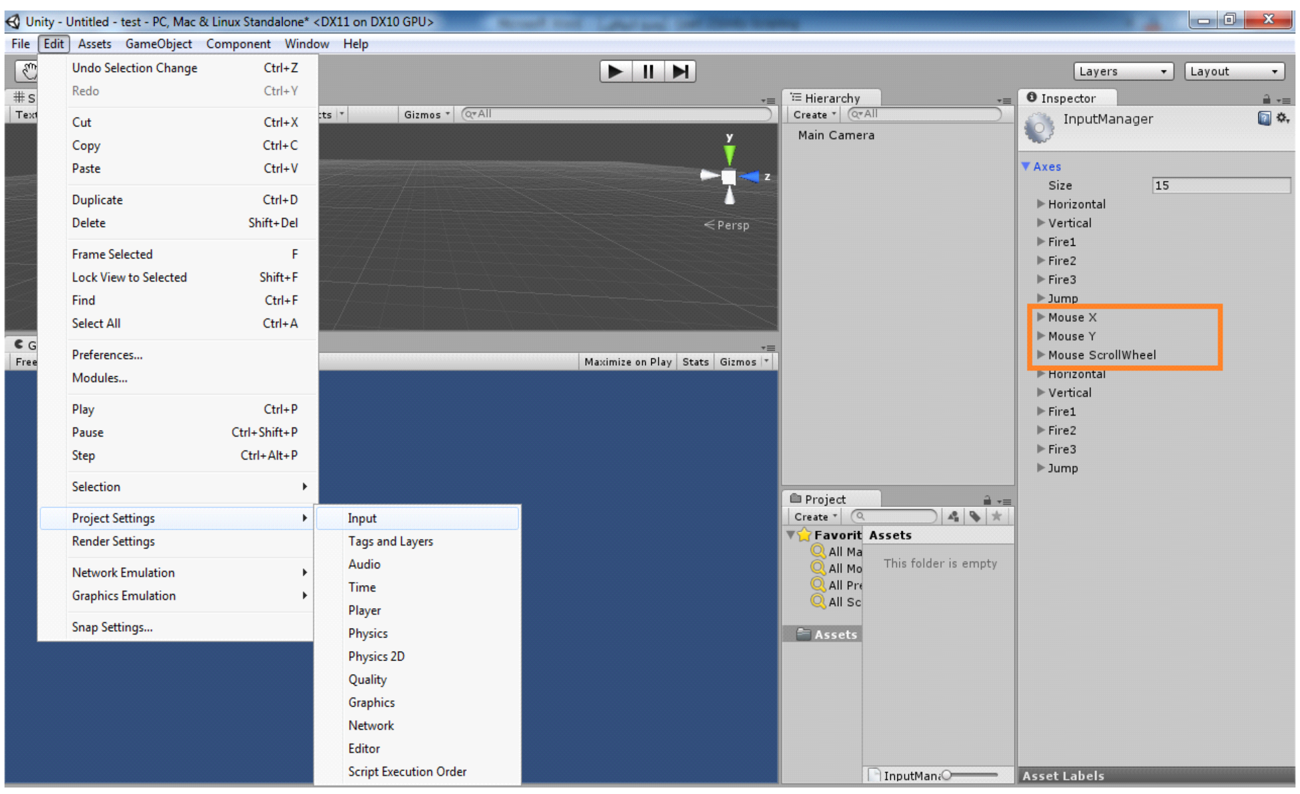
الان تلاحظ في السطر 6 اننا قمنا بإضافة الـ transform و هي التي تتحكم بربط الاشياء و تحولها و ما الى ذألك , و في السرط 7 قمنا بإضافة سرعة لهذا الكائن , و في السطر 16 قمنا بإضافة نوع الازاحة و كما تلاحظ اننا وضعنا السرعة داخل القوسين , تجنبنا ال up و ال forward , المهم في السطر 17 هنا قمنا كتابة ال transform و هذه المرة استدعينا ال LookAt معنا ان هذا الكائن سيقوم با النظر الى نقطة معينة او شيء معين يتم تحديده , في السطر 6معنا متغير اسمة Player هذا ال Player راح نضيفه الى ال LookAt و لكي نقوم بسحب هذا الكائن الذي نريد ان نقوم بتدويره حولة...



تلاحظ اننا قمنا بسحب ملف الاسكربت الى الكائن Player و قمنا بسحب الىGame الى داخل Player بالضبط في transform.LookAtو الان مع تشغيل البرنامج راح تلاحظ ان الكائن Player يدور حول الكائن المربوط Game في البرنامج .

هنا بالنسبة للمتغير راح نتكلم علية اكثر.
نحاول نجعل هذا الكائن ينظر الى مؤشر الماوس , هنا نفس الشيء راح نستخدم متغير , transform.LookAt لكن السؤال هنا كيف راح توصل الى مؤشر الماوس؟
نرجع الى شروحات سابقة لطريقة الادخال بالنسبة لمفاتيح الكيبورد استخدمت متغير ادخال Input علشان نوصل الى مؤشر الماوس

اولاً من Edit >Project Settings >Input في الصورة اسفل راح تلاحظ انني حددت على موقع الماوس بالنسبة للمحورين y,x و عجلة الماوس ان صحت العبارة , هنا بما ان الماوس ضمن الادخالات في اليونتي , راح نحدد موقع الماوس في المتغير transform.LookAtبشكل مباشر باستخدام متغير الادخال Input :



بشكل مباشر في اليونتي اكتب السطر 10:

```
Test.cs
Test ▸ Update ()
1 using UnityEngine;
2 using System.Collections;
3
4 public class Test : MonoBehaviour {
5
6
7
8     void Update () {
9
10        transform.LookAt (Input.mousePosition);
11    }
12 }
13
```

هنا انا حددت موقع الماوس ووصلت له عبر متغير الادخال Input و MousePosition تعني موقع الماوس في البرنامج, اينما كان موقع الماوس في البرنامج راح يظل ينظر الية الكائن.

بالنسبة للتحكم بموقع الماوس على احد المحاور يعني مثلاً انا اريد هذا اكائن ينظر الى موقع الماوس على محور x ... تابع

في نفس الملف اكتب السطر 11

```
Test.cs
Test ▸ Update ()
1 using UnityEngine;
2 using System.Collections;
3
4 public class Test : MonoBehaviour {
5
6
7
8     void Update () {
9
10        transform.LookAt (Input.mousePosition);
11        transform.eulerAngles = new Vector3 (transform.eulerAngles.x, 0, 0);
12    }
13 }
14
```

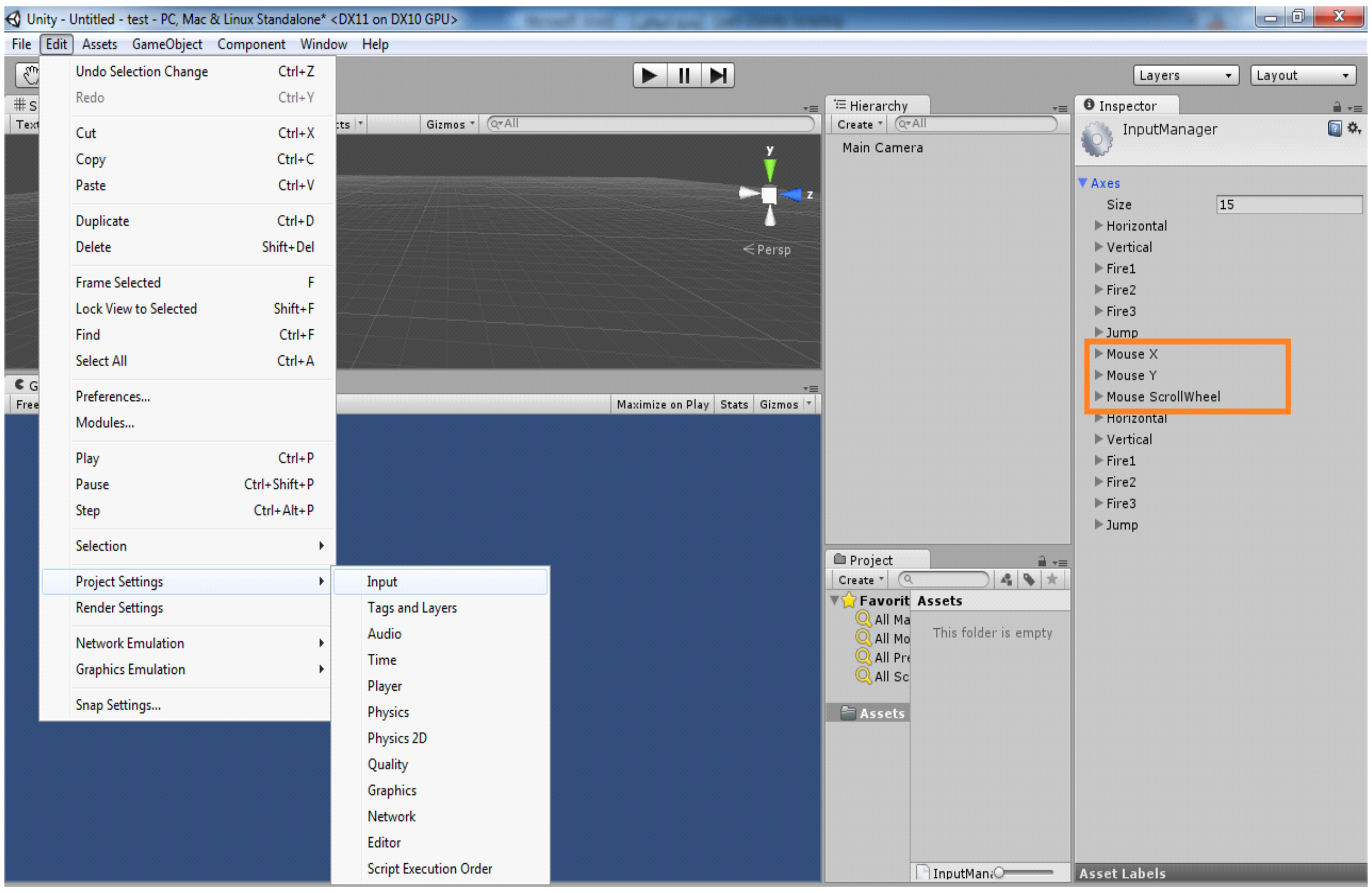
للمتغير transform.eulerAngles هو الي راح يخلي هذا الكائن يتحرك على محور معين وعشان نوصل الى المحاور الثلاثة x,y,z راح نستخدم Vector3 هو الي راح يوصلنا الى المحاور الثلاثة مثل ما استخدمناه في Translate و Rotate هنا عند فتح القوسين راح يطلب منك تحدد الثلاثة المحاور و انت راح تكتب المتغير transform.eulerAngles وراح تحدد نوع امحور كما هو موضح .

الان جرب تكتب هذا الكود و شغل البرنامج و راح تلاحظ ان الكائن ينظر الى المؤشر الى محور x فقط كيف راح تعرف هذا ... من Transformفي البرنامج راح تلاحظ ان قيمة المحور x هي الي تتغير فقط...

شرح مفاهيم في GetAxis

التعامل مع الماوس"Mouse"

في الشرح السابقة تطرقنا الى التحكم بالماوس و كيف نوصل الية و عرفنا اشياء كثيرة , هنا راح نزيد نتعمق بالتحكم بالماوس و راح نعمل اشياء تساعدنا لفهم كيف نتعامل مع الماوس .
نرجع الى صورة السابقة :



هنا راح نتعامل مع موقع الماوس بالنسبة للمحورين و كيف راح يعملوا معنا...
هنا معنا وقع الماوس بالنسبة لمحور x و الي هو (Mouse X) نفس الشيء على محور ,(Mouse Y) هنا السؤال كيف نوصل الية في البرمجة ؟ ... تابع

Test.cs

No selection

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {

    void Update() {

        Input.GetAxis ("Mouse X");
        Input.GetAxis ("Mouse Y");
    }
}
```

لاحظ في الصورة انني استخدمت GetAxis هنا هو يطلب مننا تحديد اسم مفتاح الادخال . من نفس القائمة السابقة راح تلاحظ العديد من المفاتيح الغير مكتوبة على شكل متغيرات لكن هي تستخدم عن طريق كتابة اسمها , هنا كتبت اسم موقع الماوس بالنسبة للمحورين x,y كما هو موضح في الصورة , لكن هنا ما راح يكون لة عمل في البرنامج بهذه الطريقة...
طيب الان عشان نوضح اكثر طريقة عمل الماوس على احد المواقع او على الموقعين في نفس الوقت , راح نجعل مكعب يدور حول نفسة على محور x اذا حركنا الماوس على محور x و نفس الشيء على محور ... y تابع

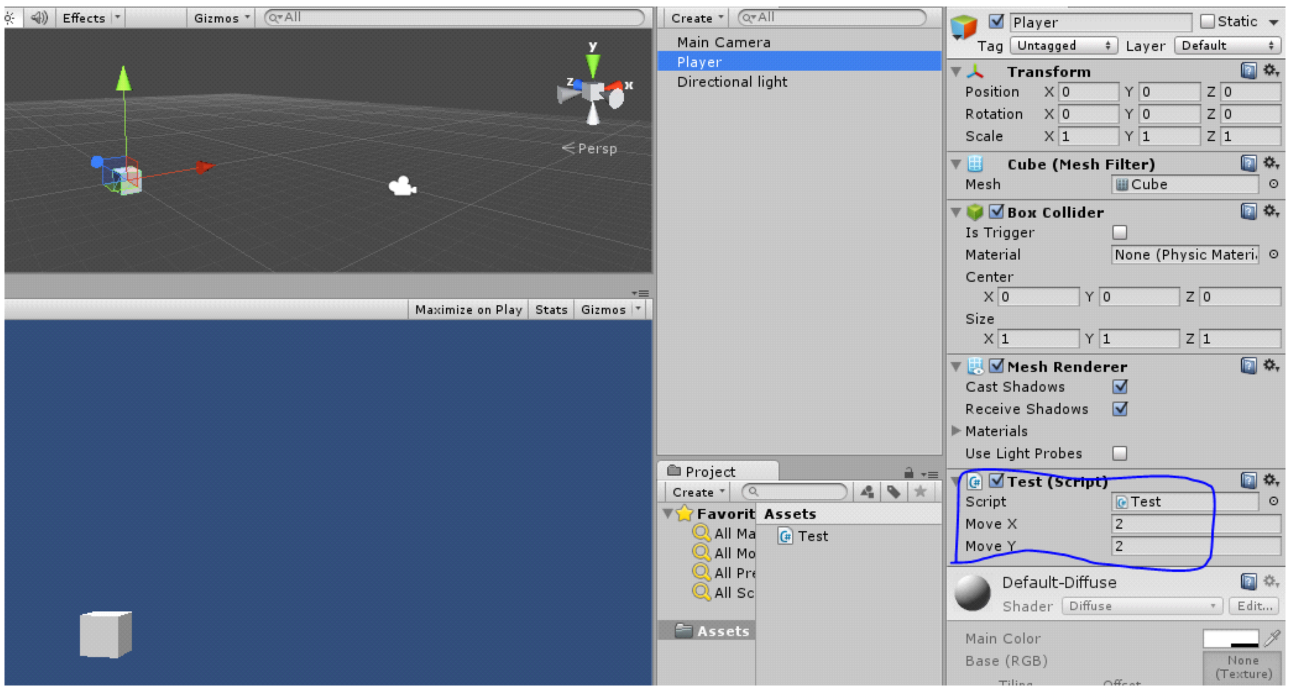
Test.cs

Test ▸ Update ()

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Test : MonoBehaviour {
5
6     public float MoveX = 2;
7     public float MoveY = 2;
8
9     void Update()
10     {
11         float MouseX = Input.GetAxis ("Mouse X") * MoveX;
12         float MouseY= Input.GetAxis ("Mouse Y") * MoveY;
13         transform.Rotate (MouseY, MouseX, 0);
14     }
15
16 }
17
```

هنا في السطر 6,7 عرفنا متغيرين راح يعطيانا سرعة على محوري X,Y و راح نربط السرعة في حركة الماوس و راح يخلق لنا حركة على المحورين اذا حرك الماوس بشكل متواصل .

في السطر 11,12 هنا كتبنا متغير من نوع float اي ان حركة الماوس على المحورين راح يكون لها قيمة معينة راح تزيد وراح تنقص في البرمجة بدون ان تراها الى ان الملف راح يظل يسحبها , هنا انا قمت بضرب المتغير الذان يحملان السرعة في موقعي الماوس ,الان عشان نعمل حدث معين راح نجعل المكعب يدور حول نفسة باستخدام متغير transform.Rotate هنا محور Y راح نوضعه على محور X و محور X على محور Y كما هو موضح في السطر 13 .



في Player راح تلاحظ ان المتغيران ظهرا , هنا راح نقدر تحكم بحركة المكعب على المحورين, مثلاً اذا حولت قيمة المحور X الى صفر هنا ما راح يتحرك ابداً, قبل كل شيء شغل البرنامج و حرك الماوس و راح تلاحظ ان المكعب يدور حول نفسة مع اتجاه الماوس على المحورين.

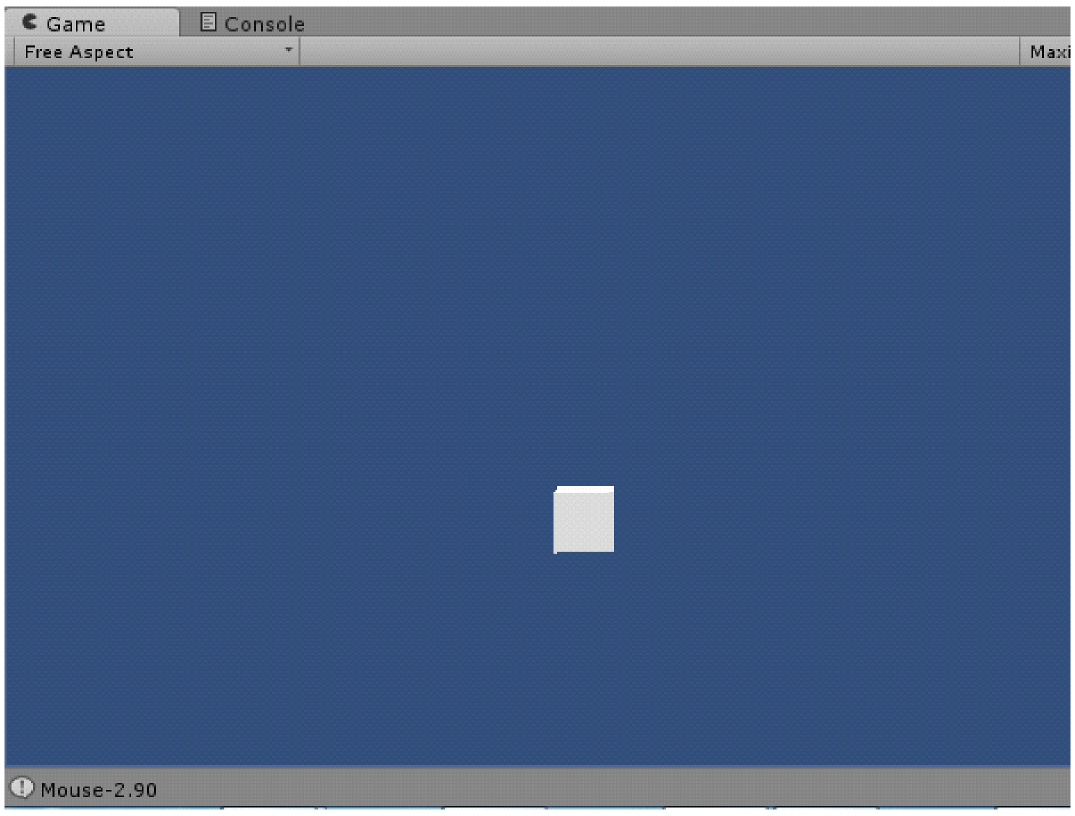
يبقى شيء واحد انا قلت سابقاً ان البرنامج يقوم بحساب حركة الماوس بشكل رقمي داخل البرمجة بدون ان تراها , هنا عشان تتأكد من ان البرنامج فعلاً يقوم بعمل هذه العملية هنا راح نقوم بطبع الاحداث الي تحصل داخل البرمجة ...

Test.cs

Test ▸ Update ()

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Test : MonoBehaviour {
5
6     public float MoveX = 2;
7     public float MoveY = 2;
8
9     void Update()
10    {
11        float MouseX = Input.GetAxis ("Mouse X") * MoveX;
12        float MouseY= Input.GetAxis ("Mouse Y") * MoveY;
13        transform.Rotate (MouseY, MouseX, 0);
14
15        Debug.Log ("Mouse" + MouseX  +MouseY);
16    }
17
18 }
19
```

لاحظ في السطر 15,16 هنا قمنا بكتابة متغير راح يقوم بطبع الاحداث لنا بعكس متغير Print هنا في القوسين راح يحدد لنا اسم المحور و راح يعطينا القيمة الي تظهر عند تحريك الماوس في الكونسول...



لاحظ في الاسفل (الكونسول) هو بيعطيك قيم تحرك الماوس على المحورين.

طيب نحاول نتحكم بموقع الكميرا عن طريق الماوس بالنسبة للمحورين x,y ؟
لاحظ الكود التالي :

test.cs

test ▸ Update ()

```
using UnityEngine;
using System.Collections;

public class test : MonoBehaviour {


    void Update () {

        transform.position += new Vector3(Input.GetAxis ("Mouse X"),Input.GetAxis ("Mouse Y"),0);

    }
}
```

كدا انت لما راح تشغل البرنامج راح تلاحظ ان الكميرا تتحرك مع الماوس ، علشان تختبر هذا الكلام ضع مكعب امام الكميرا علشان تلاحظ ان الكميرا تتحرك مع الماوس ، الان بنفس الطريقة علشان نقدر نتحكم بالماوس بشكل كامل ،نحاول نتحكم بعجل الماوس ، نضعها على محور z علشان نقدر نقرب الى المكعب و نبعد منة عبر العجلة ... تابع في نفس السطر :

test.cs

test ▸ Update ()

```
using UnityEngine;
using System.Collections;

public class test : MonoBehaviour {

    // Update is called once per frame
    void Update () {

        transform.position += new Vector3(Input.GetAxis ("Mouse X"),
                                          Input.GetAxis ("Mouse Y"),
                                          Input.GetAxis ("Mouse ScrollWheel"));

    }
}
```

لاحظ المحور الاخير ضعنا فية اسم عجلة الماوس في اليونتي ، هنا عند تشغيل البرنامج قم بتدوير عجلة الماوس و راح تلاحظ انك تقرب من المكعب .

بالنسبة للـ Horizontal و Vertical بشكل عام هم الي راح يعملوا حركة لهذا الكائن بشكل عمودي و افقي ...

شرح متغير Vector3

بالنسبة لمتغير Vector3 عرفنا ان ادا المتغير الي بتتخزن فية المحاور الثلاثة (x,y,z) و دائماً لما نريد نوصل لهم نستخدم متغير Vector3، في البرمجة نستخدمة بهذا الشكل new Vector3 اي اننا نستدعي Vector3 جديد في الكود و دائماً القيم الي نطرحها في Vector3 ما تتغير الا من البرمجة نفسة ، صحيح انت تقدرتعمل متغير من نوع float او int و تغير قيمتة ، بس هنا راح تعرف كيف تغير القيم و على اي محور تريدة في Vector3 من البرنامج و ليس البرمجة

في public راح نعرف متغير جديد من نوع Vector3 و في الكود راح نستدعية بأسمة ...لاحظ الكود التالي :

```
using UnityEngine;
using System.Collections;

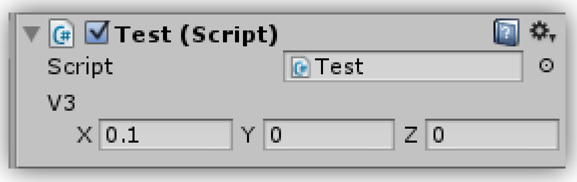
public class Test : MonoBehaviour {

    public Vector3 V3;

    void Update () {

        transform.localScale += V3 ;
    }
}
```

لاحظ هنا انني عرفت المتغير Vector3 و في الكود استدعيتة بأسمة V3 ، الان نرجع للبرنامج ، لاحظ ما سيظهر :



الان ظهرت المحاور الثلاثة ، طبعاً ادا كلة من المتغير Vector3 هنا فيك تكتب اي قيمة بشكل عادي دون وضع اي حروف و راح يعمل معك بشكل جيد .
ادا كنت تريد هذا المتغير يكون خاص في البرمجة اي انة ما يظهر في البرنامج، راح تغير خصوصية المتغير من عام الى خاص (private) طبعاً هنا راح تخزن قيم المتغير في المكان نفسة لاحظ :

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {

    private Vector3 V3 = new Vector3 (0.1f,0,0);

    void Update () {

        transform.localScale += V3 ;
    }
}
```

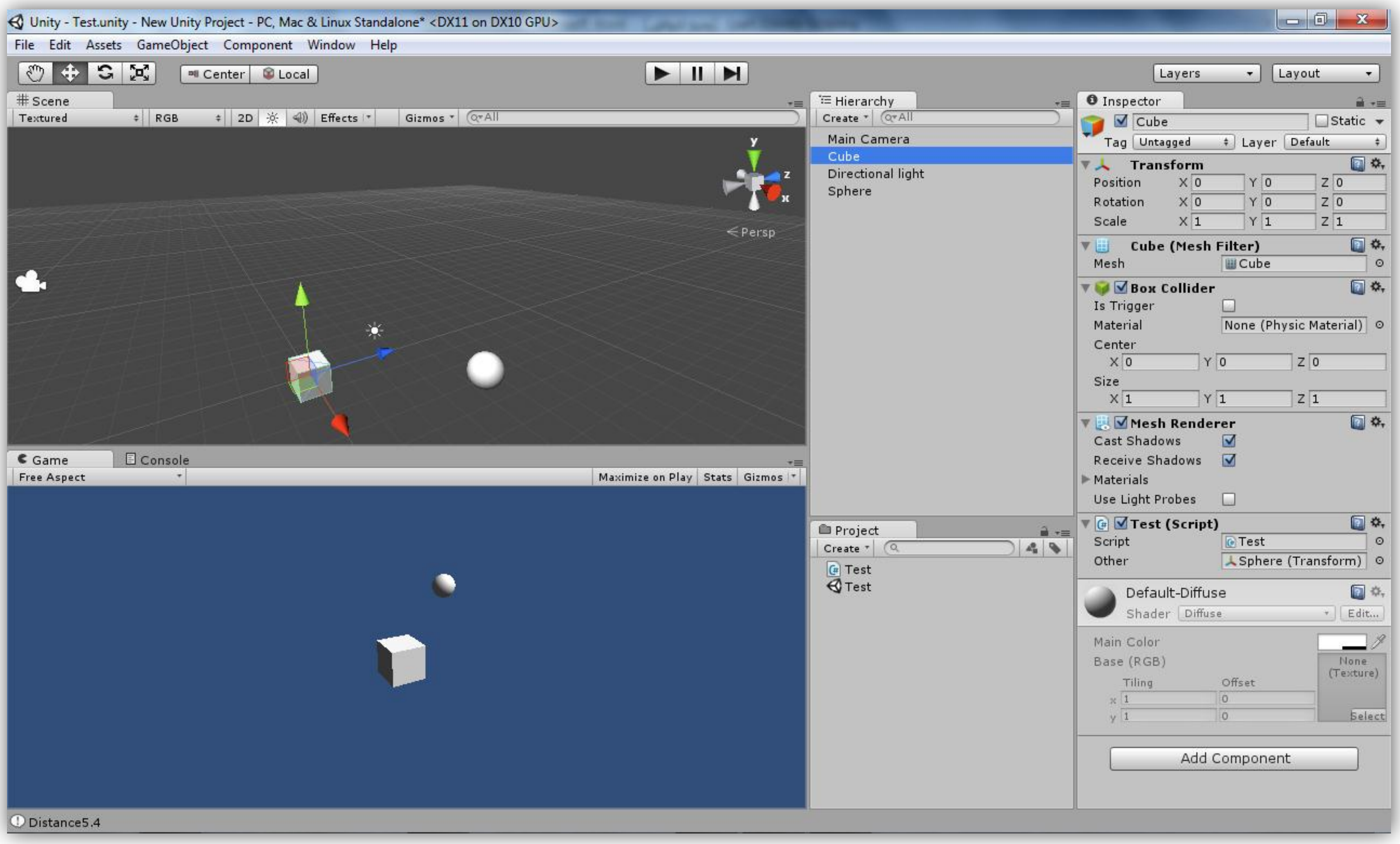
لاحظ انني وضعت قيمة في المتغير نفسة لكن ان رجعت للبرنامج ماراح تحصل المحاور الثلاثة و في النهاية فيك تستخدم الطريقة الي تناسبك و تساعدك في العمل بسرعة ☺.

متغير Vector3.Distance :

المتغير دا (Distance) هو الي بيحسب لنا بعد كائنين عن بعض او بمعنى آخر هو الي بيحسب المسافة بين اي كائنين او اكثر ، طيب انت علشان تستخدم هذا المتغير في البداية راح نعمل متغير transform علشان نعمل مسافة بين كأئنين ، لاحظ الكود التالي :

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Test : MonoBehaviour {
5
6      public Transform Other;
7
8      void Update () {
9
10         float dist = Vector3.Distance (Other.position, transform.position);
11         Debug.Log ("Distance" + dist);
12     }
13 }
```

هنا عملت متغير من نوع Transform علشان نخزن فية الكائن الي راح نعمل مسافة معة ، في السطر ١٠ لاحظ انني عرفت متغير من نوع float علشان يحسب المسافة الي بين الكائنين بصيغة عددية ، تلاحظ انني كتبت المتغير Other و حددت ان المسافة راح تكون في position تبعة و الفاصلة الثانية تطلب منك تحدد مسافة الكائن الثاني على اي موقع ، افيك تعرف متغير Transform او انك تستخدم الكائن تبعك ، هنا راح استخدم الكائن تبعي و بشكل مباشر راح احدد موقعة في position الي هو (transform.position) كدا راح يحسب المسافة بينهم ، هنا اعمل طبع علشان تعرف كم المسافة بينهم ، نرجع الى البرنامج راح تحصل المتغير تبعنا موجود ، الان قم بسحب الكائن المراد الية و شغل البرنامج و راح يتم حساب المسافة ...



شرح متغير Vector3.back :

متغير back سهل جداً ما يحتاج شرح مطول لان طريقة استخدامة سهله ،هو راح يعمل لهذا الكائن حركة الى الخلف ، طريقة استخدامة بهذا الشكل :

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {


    void Update() {

        transform.position += Vector3.back * Time.deltaTime;
    }
}
```

بمجرد وضعة الملف في اي كائن راح يتراجع الى الخلف تلقائياً ...

ايضاً متغير forward بيعمل لنا حركة الى الامام فقط غير كلمة back الى forward و راح يعمل معك ، متغير up بيعمل لنا حركة الى الاعلى و متغير down بيعمل لنا حركة الى الاسفل ، ايضاً متغيري right , left يعملوا حركة جانبية .

شرح متغيري Vector3.Max , Vector3.Min :

متغيري Max , Min هم الي يحددوا لنا اعلى و اقل قيم لـ Vector3 : انت علشان تختبر هذا الكلام بنحاول نعمل طبع لهذه القيم ، لاحظ الكود التالي :

```
using UnityEngine;
using System.Collections;

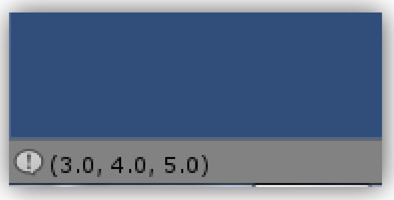
public class Test : MonoBehaviour {

    public Vector3 MaxVector3 = new Vector3(3,4,5);
    public Vector3 MinVector3 = new Vector3(0,1,2);


    void Update() {

        print (Vector3.Max (MaxVector3, MinVector3));
    }
}
```

لاحظ في البداية اننا عرفت متغيري Vector3 و خزنت في الاول اعلى قيم و في الثاني اقل قيم ، في Update لاحظ اننا طبعت الكلام كدا و حددت ان هذا الـVector3 بيعطينا اعلى قيم من ضمن المتغيرين ، في البرنامج لاحظ انه يطبع لي اعلى قيم :



و نفس الشيء في متغير Min لكن متغير Min يطبع لي اقل قيم ما بين المتغيرين ...

شرح متغير Vector3.Scale :

متغير Vector3.Scale بسيط جداً ، الفائدة منة انة يقوم بضرب الارقام التي في نفس المحاور مع بعضها ،كيف تستخدم هذا المتغير ؟ لاحظ الكود التالي :

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {


    void Update () {
        print(Vector3.Scale(new Vector3(1, 2, 3), new Vector3(2, 3, 4)));
    }
}
```

انت لما راح تكتب Vector3.Scale هو بيطلب منك متغير a من نوع Vector3 و متغير b من نوع Vector3، هنا بشكل مباشر راح تفتح القوسين و راح تعمل المتغيرين و بتكتب القيم التي تريدها ، في الكود دا لاحظ انة راح يقوم بضرب الارقام التي في نفس المحاور اي ان هذا الكود راح تكون قيمة بهذا الشكل : (12 ,6 ,2)، انت اكتب اي ارقام ثانية و راح تظهر النتيجة بشكل مباشر .

متغيرى Vector3.one , Vector3.zero :

متغيرى Vector3.one , Vector3.zero وظيفتهم انهم راح يساعدانك في استرجاع موقع الكائن الرئيسي في البرنامج ، متغير Vector3.zero راح يقوم بأرجاع الكائن الى النقطة (0,0,0) في position أي انة راح يرجع الكائن الى نقطة الوسط ، اما متغير Vector3.one راح يضيف رقم واحد الى المحاور الثلاثة (1,1,1) بهذا الشكل ، طيب هنا كيف تستخدمه في البرمجة ؟ لاحظ الكود التالي :

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {

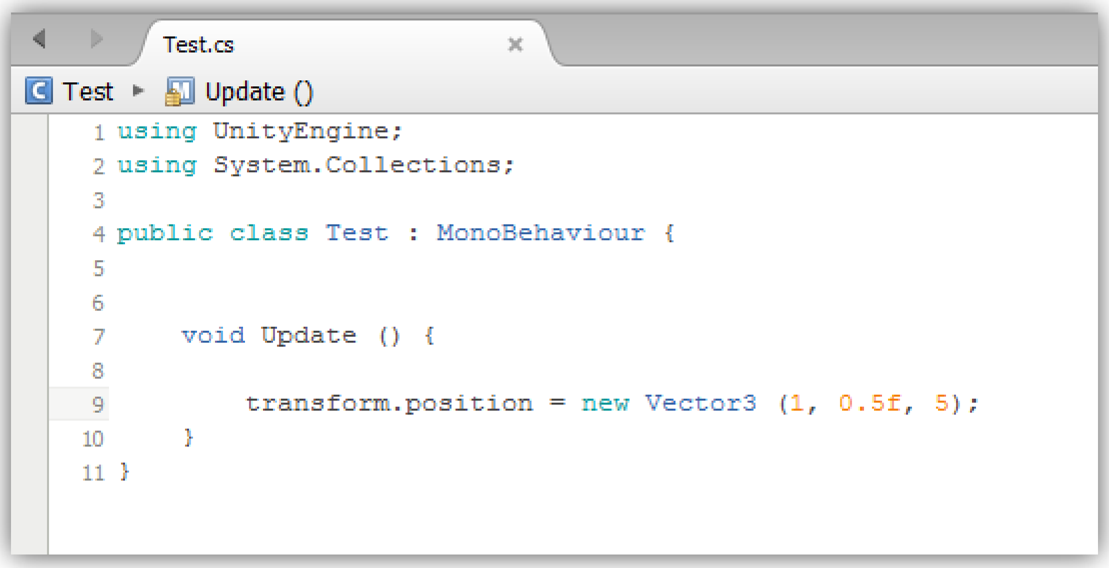

    void Update () {
        transform.position = Vector3.one;
    }
}
```

الى هنا نكون قد وصلنا الى نهاية شرح متغيرات Vector3، الان عليك ان تستكشف باقي متغيرات Vector3 و راح تتعلمها بسهولة ☺.

شرح بعض متغيرات Transform

صحيح ان متغير transform مفهوم و هو اكثر متغير استخدام في البرمجة ، هنا راح نشرح بعض المتغيرات فية و التي راح تساعدنا في العمل ، اول متغير راح اشرحة هو متغير transform.position متغير transform.position هو الي بيحدد لنا موقع الكائن في البرنامج على محاور الـ Position ، انت كيف فيك تستخدم المتغير اذا علشان تخلف نوع من الحركة او حدود معينة او اي شيئ آخر ، راح نبدأ الشرح تدريجياً من البداية الى ان نصل الى معنى مفهوم transform.position .

في البداية راح نكتب متغير transform.position بصيغتة الطبيعية و راح نولد نقلة الى مكان آخر ، اي راح نجعل هذا الكائن ينتقل الى مكان آخر من موقع في الـ position ... لاحظ الكود التالي :



```
using UnityEngine;
using System.Collections;

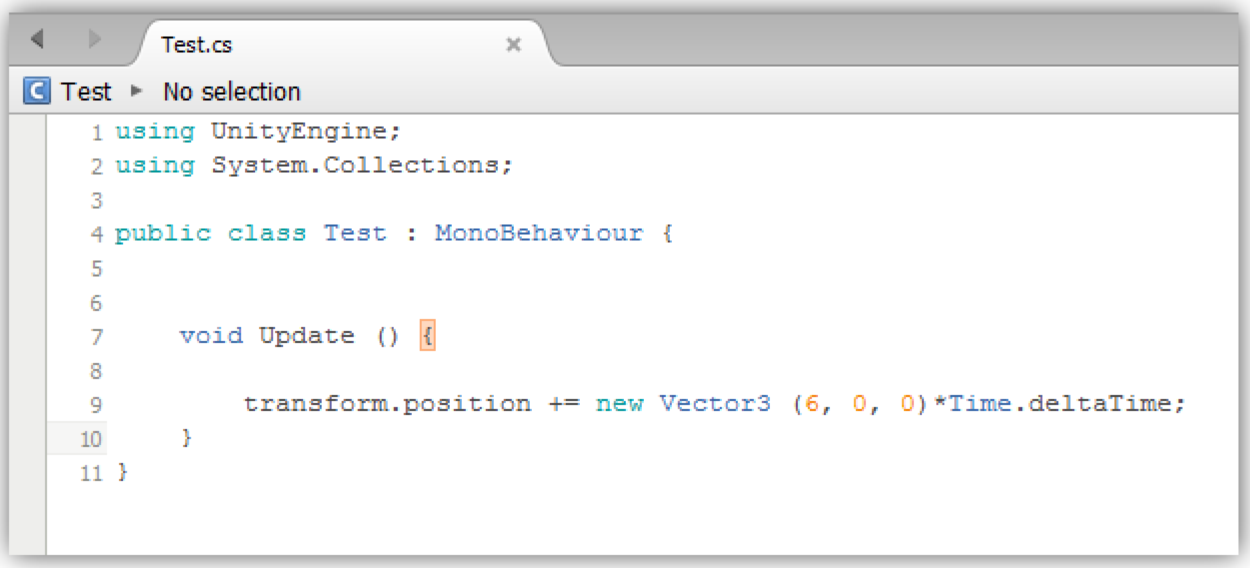
public class Test : MonoBehaviour {


    void Update () {

        transform.position = new Vector3 (1, 0.5f, 5);
    }
}
```

لاحظ هنا الكائن راح ينتقل الى هذة الارقام في الـ position راح تحصل النقلة بسرعة بس ماراح يكون لها تحديث اي ان هذه القيمة ثابتة ، راح يوصل لها الكائن و يتوقف ، طيب انت كيف فيك تعمل تحديث لهذة الحركة ؟؟

نفس الكود فقط غير = الى =+ كذا راح يعمل تحديث ، بس هنا راح تلاحظ ان الحركة صارت على كل المحاور ، هنا انت حدد رقم على محور واحد و اضربة في (فارق الوقت) ،لاحظ الكود التالي :



```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {


    void Update () {

        transform.position += new Vector3 (6, 0, 0)*Time.deltaTime;
    }
}
```

لاحظ هنا كيف عدلت على الكود و جعلت الحركة على محور واحد و كذا راح يضل يعمل تحديث للحركة ، طيب هنا كيف تعمل حدود للحركة ؟ يعني كيف تجعل هذا الكائن يتحرك الى مكان معين او داخل نطاق معين ؟ ... لاحظ الكود التالي :

Test.cs

Test ▸ Update ()

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Test : MonoBehaviour {
5
6
7     void Update () {
8
9         transform.position += new Vector3 (Input.GetAxis ("Horizontal"), 0, 0) * Time.deltaTime * 6;
10
11        if(transform.position.x >= 5f)
12        {
13            transform.position = new Vector3 (5, 0, 0);
14        }
15        if(transform.position.x <= -5f)
16        {
17            transform.position = new Vector3 (-5, 0, 0);
18        }
19
20    }
21 }
```

لاحظ في السطر ٩ عرفت كود الحركة العادية على محور x بالنسبة لـ Position، في السطر ١١ الى السطر ١٨ عملت شروط علشان يعمل حدود للحركة على محور x لاحظ هنا كتبت transform.position.x لانة في الشرط بيطلب منك تحدد اسم المحور و تحدد قيمة او فيك تتركة فاضي ، ولانة متغير position راح تعمل حدود الى احد المحاور او على كل المحاور ، في السطر ١١و١٥ انا خليت اكبر قيمة يوصل اليها الكائن او اقصي مدى يوصل الية هو الرقم 5 بالنسبة للقيم السالبة و الموجبة ، في السطر ١٣و١٧ لاحظ انني كتبت الشرط تبعطي الي بيعطنا الحدود معناة انة اذا وصل الكائن الى القيمة 5 او-5 راح يتوقف عن هذه القيمة و هو الموجود في جوابي الشرط في السطري ١٣و١٧ ، و نفس الشيئ على باقي المحاور .

شرح متغير transform.eulerAngles :

عرفنا سابقاً ان Rotate هو الي يتحكم بموقع الكائن بالنسبة لـ Rotation هو الي يقوم بعمل تدوير للكائن على المحاور ، هنا معنا متغير ثاني بيعمل لنا ازاحة جزئية او دائمة لهذا الكائن على أي محور تريدة ، هذا المتغير بيطلب منك تحديد نوع المحور و من خلالة يمكنك ربط محورين مع بعض ، متغير eulerAngles قد يسهل عليك الكثير ... نتابع :

اكتب الكود في السطر ١٠ :

test.cs

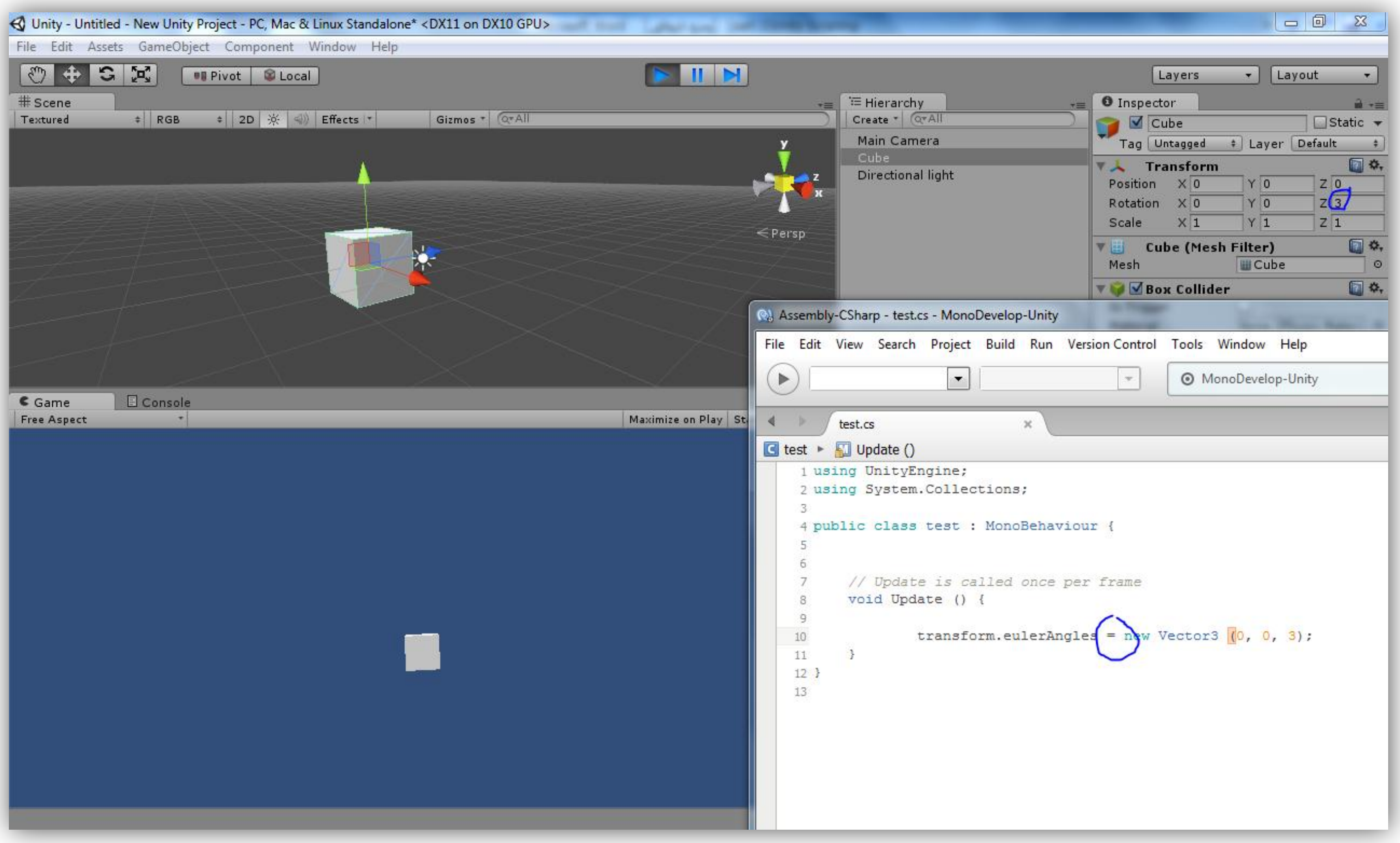
test ▸ Update ()

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class test : MonoBehaviour {
5
6
7     // Update is called once per frame
8     void Update () {
9
10            transform.eulerAngles += new Vector3 (0, 0, -3);
11    }
12 }
13
```

طيب بالنسبة لهذا الكود لما راح توضعة في مكعب او أي كائن في البرنامج راح تلاحظ ان هذا الكائن يدور حول نفسة !
لاحظ عن = انني وضعت علامة + أي ان القيمة التالية راح تزيد في هذه الازاحة بحيث انة راح تعطي لنا حركة دائمة للكائن ،

بمعنى آخر ان القيمة 3- راح تزيد في محور z ،هذا لاننا وضعنا علامة =+ قبل القيمة ، طيب لو وضعنا = بدل += في البداية وش راح يحصل ؟؟

الي راح يحصل ان هذا الكائن راح يدور الى الموقع 3- في Rotation . لاحظ الصورة التالية :



لاحظ هنا انة عند تشغيل البرنامج تحرك المكعب الى القيمة 3 على محور Z في Rotation، طيب الان قم باضافة += بدل = و راح تلاحظ ان المكعب يدور حول نفسة على محور z .

نحول نعمل شيئ بسيط يوضح طريقة عمل متغير eulerAngles أكثر ، لاحظ في الكود التالي :

```
test.cs
test ▸ Update ()

using UnityEngine;
using System.Collections;

public class test : MonoBehaviour {


	// Update is called once per frame
	void Update () {

		if(Input.GetAxis ("Horizontal") > 0)
			transform.eulerAngles += new Vector3 (0, 0,-3);

		if(Input.GetAxis ("Horizontal") < 0)
			transform.eulerAngles += new Vector3 (0, 0,3);
	}
}
```

تلاحظ اني عرفت شرطين في البرمجة ، طبعاً لما اكتب قوسين المجموعة {} لانة يوجد كود في كل شرط ، المهم هنا بالنسبة للـ Horizontal هنا عندما تكون قيمة اكبر من الصفر راح يتحرك الى اليمين و اذا كانت اقل من الصفر راح يتحرك الى اليسار ، كيف راح تتحكم بالقيم ؟ راح تتحكم فيها عن طريق مفتاحي a,d او السهمين اليمين و اليسار ...

قم بوضع هذا الكود في مكعب و راح تلاحظ الفرق ،في النهاية متغير eulerAngles هو الافضل استخداماً في عمل الإزاحات ، وراح يساعدك في عمل حركة جانبية للكائن و ما الى ذالك ...

شرح متغير transform.localScale :

طبعاً عرفنا سابقاً ان Scele هو الي يتحكم بحجم الكائن في البرنامج ، طيب هنا عشان توصل لـ Scele في البرمجة راح تستخدم متغير منوع transform الي هو localScale وهذا الي راح يتحكم بحجم الكائن في البرنامج عن طريق البرمجة ، طيب هنا علشان توصل لهذا المتغير راح راح تستدعية من متغير transform ... لاحظ الكود التالي :

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {


    void Update () {

        transform.localScale += new Vector3 (0.1f, 0.1f, 0.1f) * Time.deltaTime;
    }
}
```

لاحظ هنا انني استدعيت متغير localScale من متغير transform ، هنا لاحظ الاشارات استخدمت =+ هو الي راح يزيد حجم الكائن بنسبة 0.1f نسبة كافية علشان تلاحظ مقدار تغير حجم الكائن ، هنا طبعاً علشان توصل للمحاور الثلاثة x,y,z بتستخدم متغير من نوع Vector3 و هذا المخزن فية المحاور الثلاثة ... الان اسحب الكود الى المكعب ولاحظ مقدار تغير حجمة في البرنامج .

ملاحظة : سبب كتابة حرف f بعد 0.1 هو انة في البرمجة لما انت تريد كتابة قيمة كسرية (float) بتضطر انك تكتب حرف f بعد الرقم و هذا يدل على ان القيمة دي من نوع float .

شرح متغير transform.RotateAround :

متغير RotateAround متغير راح يعمل لنا حركة دورانية حول شيئ معين ... مثلاً انت تريد مكعب يدور حول كرة او يدور حول نقطة الصفر او اي شيئ استخدم متغير RotateAround ، طبعاً المتغير بيطلب منك ٣ اشياء الي هي

(Vector3 point ,Vector3 axis ,float angle) لاحظ انة بيطلب منك متغيرين من نوع Vector3 السبب ؟؟

السبب هو لان هذا المتغير بيطلب انة يستخدم محاور كائن معين وبيطلب تحديد نقطة على احد المحاور في كائن معين ، لهذا هو بيطلب متغيرين من نوع Vector3 ، في الاخير بيطلب متغير من نوع float علشان نحدد سرعة معين لهذا الدوران ، علشان تعرف اكثر لاحظ الكود التالي :

```
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {

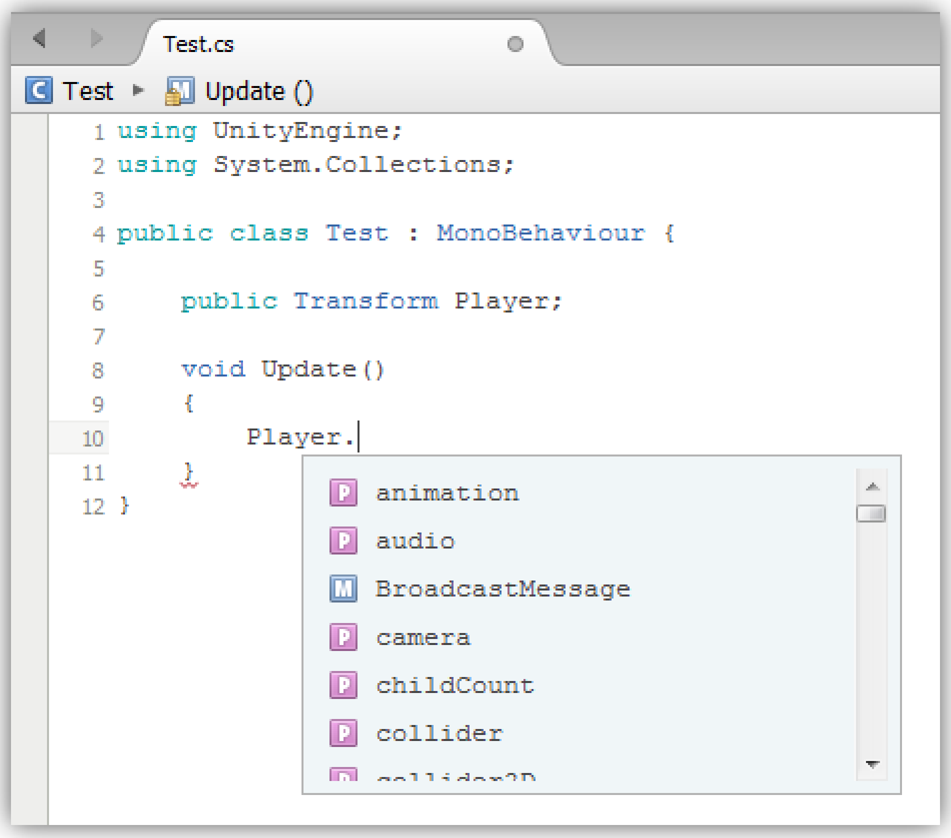

    void Update () {

        transform.RotateAround(Vector3.zero, Vector3.up, 20 * Time.deltaTime);
    }
}
```

في الكود انا عرفت ان النقطة الي راح يدور عليها الكائن هي مركز البرنامج و الحركة الدورانية تبدأ على محور x او up بعدها حددت سرعة الحركة 20 ة ضربتها في (فارق الوقت) و كدا الكود يعمل بشكل جيد ☺.

لماذا تكتب بعض المتغيرات في Transform و البعض الآخر في GameObject

بالنسبة لكتابة بعض المتغيرات في transform هو انني اريد ان اعامل هذا الكائن كمتغير من نوع transform أي انني اخزن متغيرات transform فية ، و نفس الشيئ في GameObject ، الى ان كل واحد منهم توجد لة متغيرات خاصة ... مثال



لاحظ هنا ، بما ان المتغير Player من نوع Transform راح يأخذ خواصة و متغيراتة و كل شيئ فية ، و نفس الشيئ في متغيرات GameObject ، و GameObject يدل على ان هذا المتغير يحتوي على كائن واحد بعكس متغير Transform فيك تخزن فية اكثر من كائن ، برضوا فيك تعامل متغير transform على اساس انة GameObject و العكس كيف هذا ؟ لاحظ الكود التالي :



لاحظ هنا ان المتغير Player عبارة عن متغير من نوع transform لكن انت علشان تعامل هذا المتغير على انة GameObject راح تستدعي متغير gameObject كدا راح يأخذ خواص المتغير GameObject ، نفس الشيئ في متغير GameObject لاحظ الكود التالي :

```csharp
using UnityEngine;
using System.Collections;

public class Test : MonoBehaviour {

    public GameObject Player;

    void Update () {

        Player.transform.position = Vector3.one;
    }
}
```

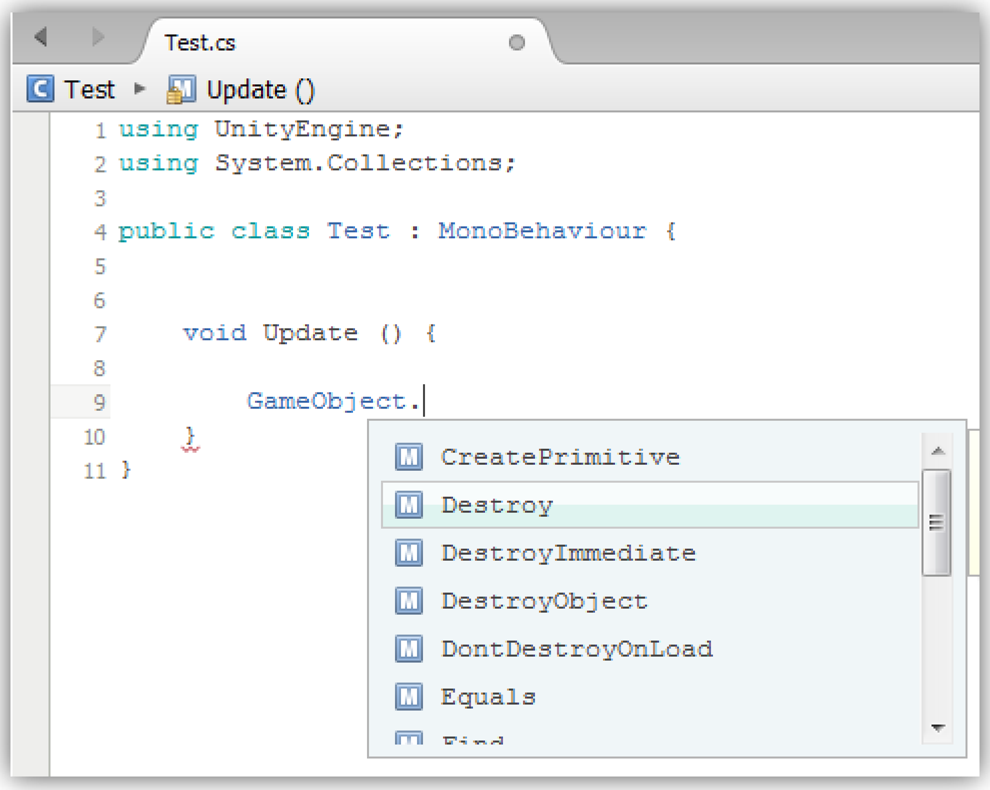
لاحظ ان نفس الشيئ نفس الكلام في الكود الاول ، حولنا متغير GameObject الى transform ...

الى هنا اترك الباقي عليك استكشف باقي متغيرات Transform بعد ما اخدت فكرة الان.☺

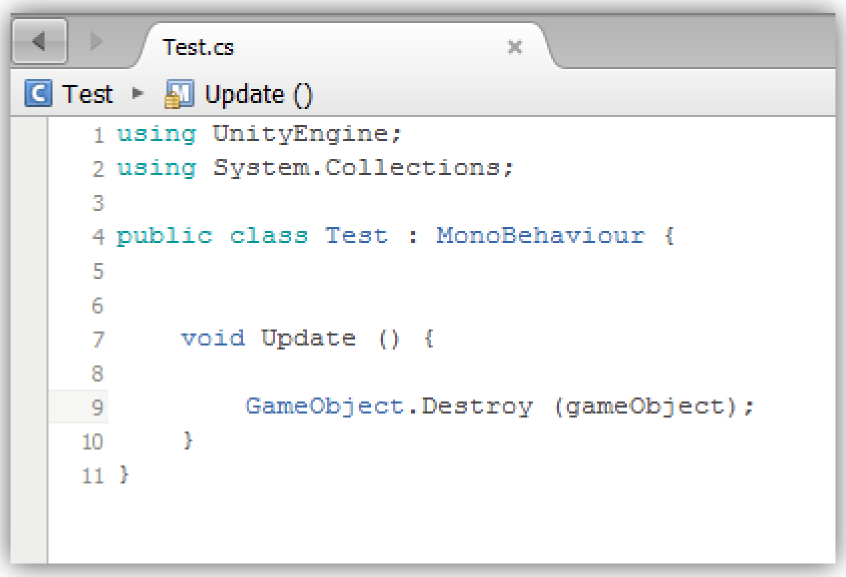
شرح بعض متغيرات GameObject

تعرفت سابقاً على متغيرات Transform وأخذت فكرة عن طريقة عملها ، هنا راح تتعرف على متغيرات GameObject و راح تتعلم كيف تتعامل معها بطريقة سليمة وصحيحة .

في البداية انت لما تكتب GameObject او تطرحة كلمة GameObject في أي مكان في البرمجة ، كدا انت بتعني ان الشروط دي بتحصل على هذا الكائن ، اما عندما تستخدم متغير من نوع GameObject كدا انت بتحدد كائن معين في البرنامج او تبحث على كائن معين ، اول متغير راح تتعرف علية هو متغير Destroy اكيد تعرف هذا المتغير بس راح نعمل شرح بسيط حولة . هنا علشان تتعرف على جميع متغيرات GameObject راح تستخدم كلاس GameObject لاحظ الصورة التالي :



لاحظ هنا كتبت الكلاس و ظهرت جميع متغيراتة و من ضمنها متغيرات Destroy المشترك مع Transform ، هنا راح نستخدم المتغير Destroy راح تفهم كيف ان هذا الكائن يتعامل مع المتغير بأسمة هو ...لاحظ الكود التالي :



لاحظ انا عن طريق هذا الكلاس استدعيت المتغير Destroy و جعلتة يدمر هذا الكائن كـ GameObject ، اما بالنسبة للـ Transform ما ان استدعيت المتغير Destroy منة ما راح تقدر تدمر الكائن كـ Transform السبب لان الكائن نفسة يتعامل في البرمجة كـ GameObject لهذا لما راح تتعامل مع الكائن نفسة او أي كائن نادية كـ GameObject .

شرح متغير GameObject.Find :

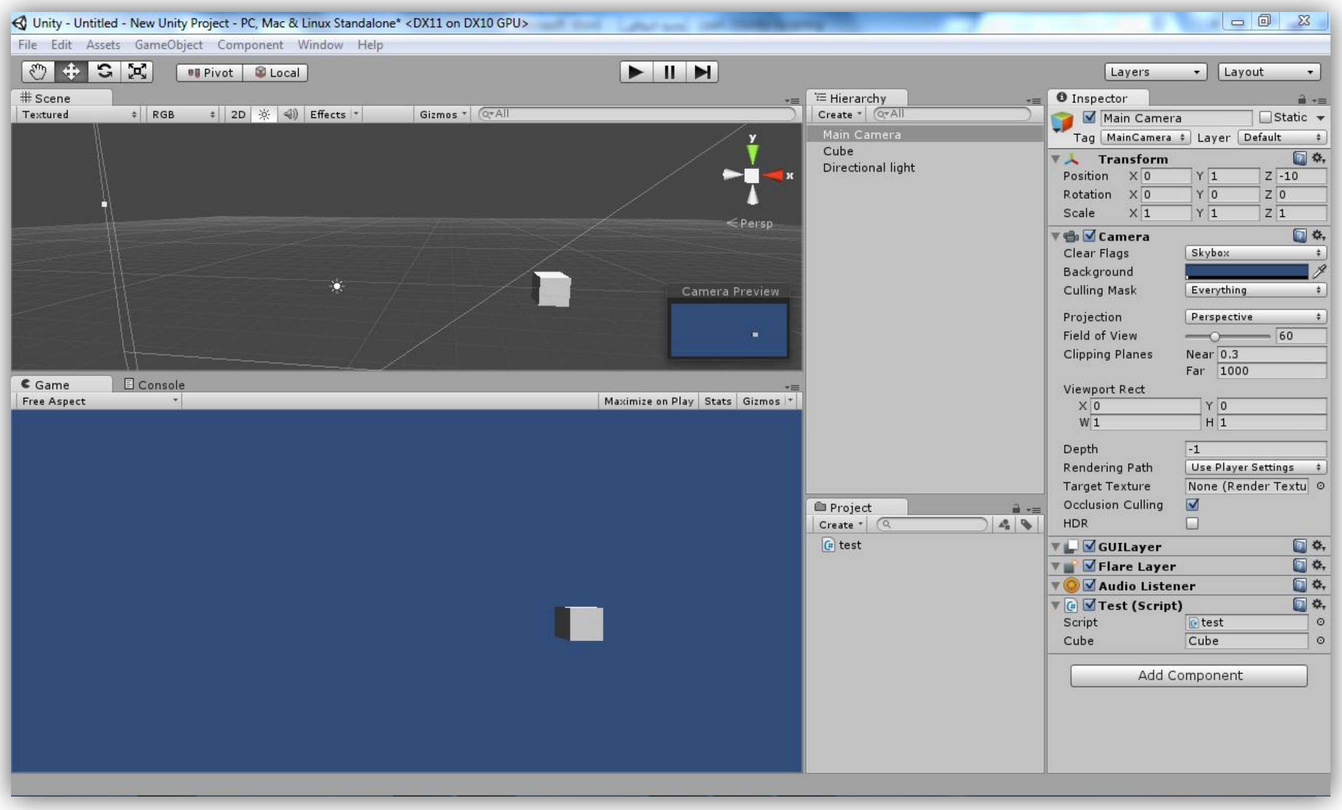
هنا اولاً راح اوضح مفهوم المتغير Find ، المتغير Find هو متغير يقوم بالبحث عن متغيرات من نوع GameObject و ليس Transform و يقوم بتخزينا او وضعها في المتغير تبعها بشكل تلقائي ، هنا قد تطرح سؤال في نفسك : ليش راح استخدم هذا المتغير دامنا استطيع سحب الكائن الى الاسكربت بنفسي ، وش راح يفيدنا هذا المتغير ؟؟

طيب انا راح اوضحلك مفهوم المتغير ، هنا انت لما يكون معك كائن prefab و فية متغير GameObject ما راح تقدر تسحب أي شيئ الى هذا المتغير في prefab لهذا المتغير اذ راح يساعدك في عمل بحث لاي كائن في البرنامج ، اكان بأسمة او بالتاج تبعة.

مثال اكتب الكود التالي :

```
test.cs
test ▸ Update ()
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6      public GameObject Cube;
7
8      void Update () {
9
10         Cube = GameObject.Find ("Cube");
11         transform.LookAt (Cube.transform);
12     }
13 }
```

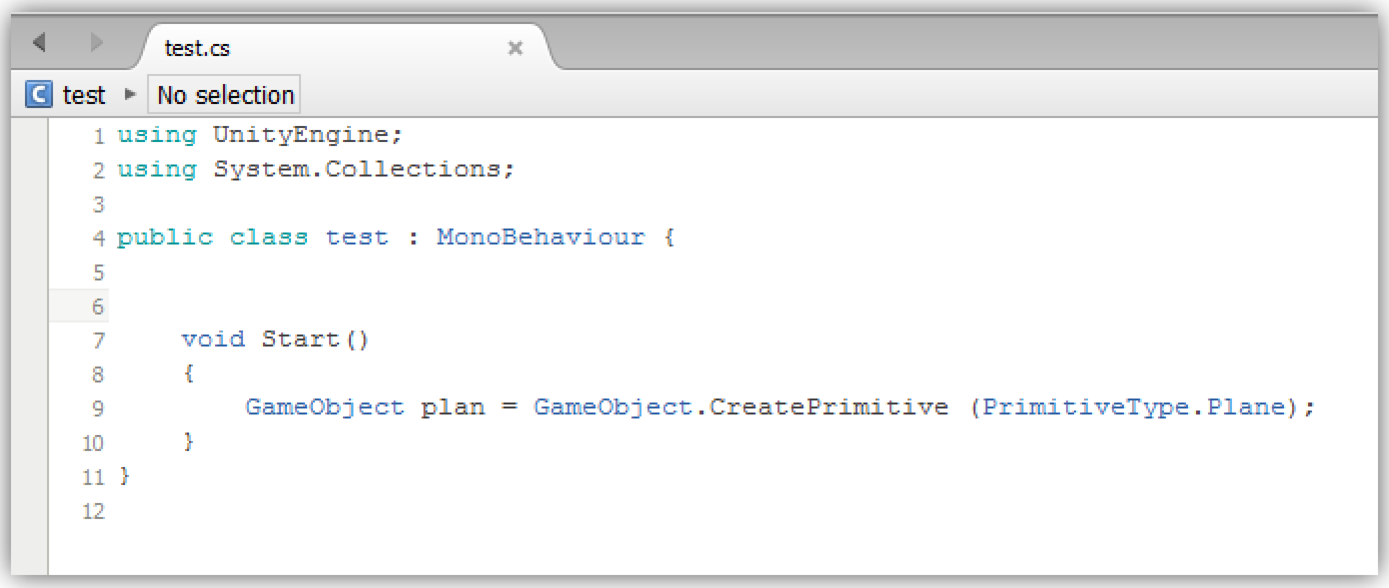
لاحظ في السطر ٦ معنا متغير من نوع GameObject بأسم Cube ، و في السطر ١٠ راح تلاحظ انني قمت بجعل هذا المتغير يبحث عن كائن بأسم Cube و يخزنة في المتغير Cube أي انة ادا وجد هذا الكائن راح يخزنة في Cube بشكل مباشر ، في السطر ١١ قمت بجعل هذا الكائن ينظر الى المكعب، لكن هنا في شيئ محتاج توضيح ، لو كتبنا أي متغير من نوع GameObject داخل متغير من نوع Transform ما راح يتقبلة راح تحصل مشكلة، هنا علشان تحول هذا المتغير الى transform نقوم بكتابة transform من متغير GameObject كما هو موضح في الصورة السابقة ، الان قم بوضع الملف في الكميرا و قم بعمل مكعب في البرنامج و الان قم بتشغيل البرنامج و راح تلاحظ ان الكميرا تنظر الى المكعب ...



بالنسبة للبحث عن كائن من التاج تبعة ، هنا راح تستدعي متغير FindGameObjectWithTag هذا امتغير الي راح يبحث عن أي كائن فية تاج معين و من نفس الطريقة السابقة قم بكتابة اسم التاج و راح تلاحظ ان الكود يعمل بشكل جيد ☺ .

شرح متغير GameObject.CreatePrimitive :

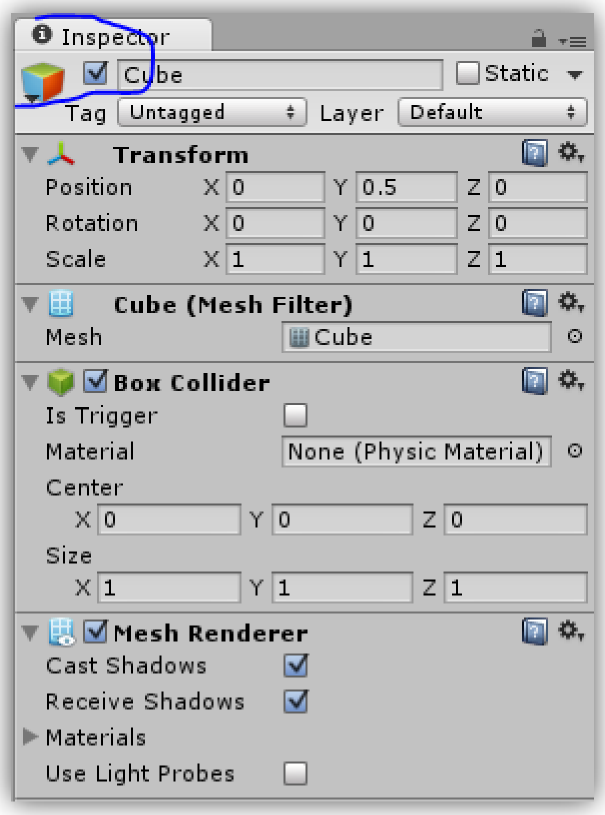
متغير CreatePrimitive بيستخرج لنا اشكال كائنات معينة و الي هي (Plane,Cube, Cylinder, Capsule, Quad, Sphere) و فيك تستخدمهم لعمل مجسمات من داخل البرمجة ، علشان تستخرج المجسمات دي بيطلب منك تستخدم متغير CreatePrimitive الي هو انشاء مجسمات،طبعاً هو احد متغيرات GameObject لان المجسمات عبارة عن كائنات تخص المتغير GameObject و لهذا خزنت فية ، الان علشان تتعرف على طريقة استخدام المتغيرCreatePrimitive لاحظ الكود :



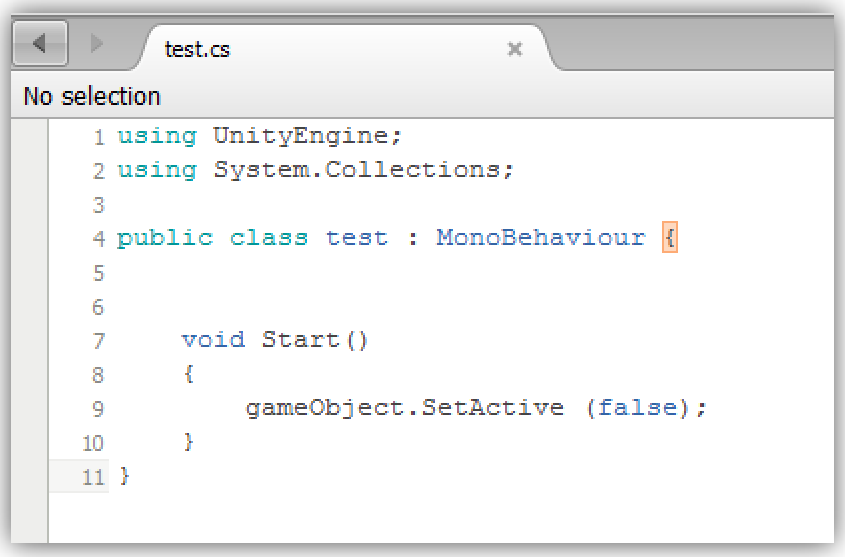
لاحظ في البداية بيطلب منك تعرف كلاس GameObject و راح تعمل لة اسم علشان يكون وسيط بين الكود و باقي الكواد ، ثانياً بتكتب المتغير كما هو موضح و في القوسين فين تستخدم احد الكائنات الستة (Plane,Cube, Cylinder, Capsule, Quad, Sphere) و كدا يكون مفهوم الـ CreatePrimitive مفهوم .

شرح متغير gameObject.SetActive :

متغير SetActive دا الي بيتحكم بأخفاء و اظهار الكائن او تفعيل و عدم تفعيل الكائن و بالنسبة لعدم تفعيل الكائن يعني ان الكائن ما راح يكون لة اي خواص يتحكم فيها ، راح يكون منفي من البرنامج ، طيب هنا كيف انا اتعرف على هذا الكلام ؟؟ نرجع الى البرنامج و لاحظ زر SetActive الي بيتحكم او تفعيل و عدم تفعيل الكائن :



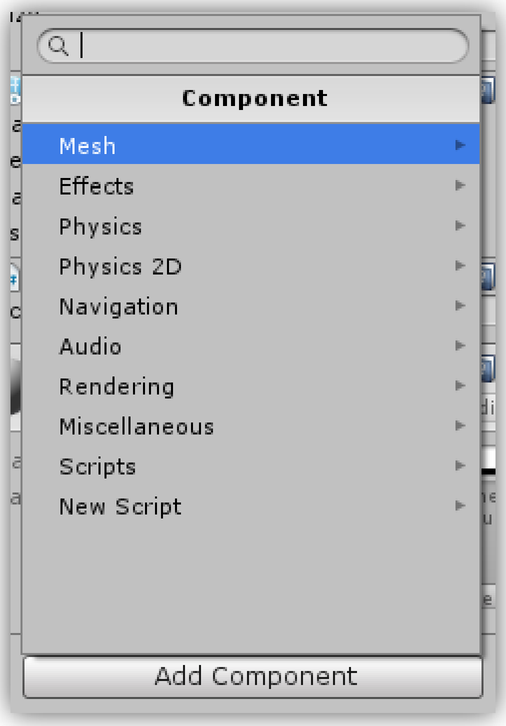
لاحظ الزر في الاعلى هذا يسمى SetActive و دا يتحكم بتفعيل وعدم تفعيل الكائن،طيب هنا تلاحظ انة يأخذ القيمة true او false اي تفعيل او عدم تفعيل ، طيب في البرمجة كيف راح تتعامل معة ؟؟ لاحظ الكود التالي :



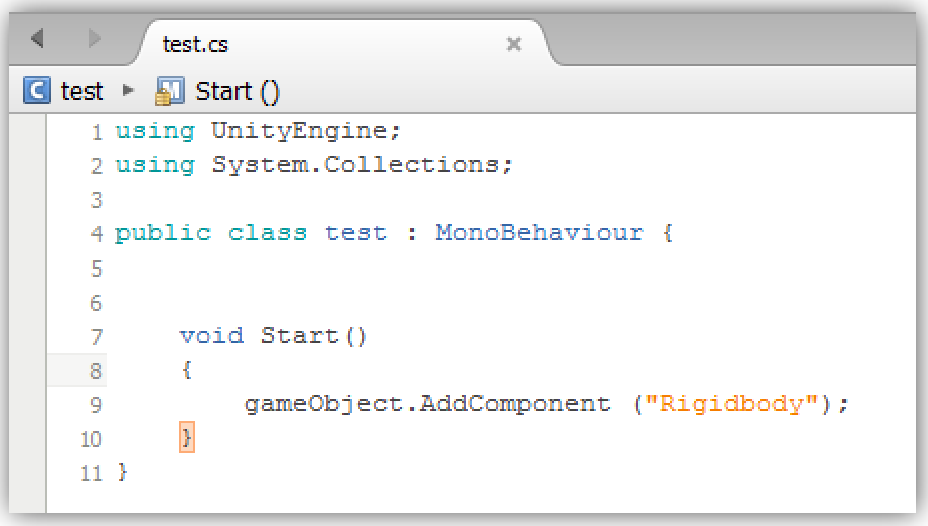
لاحظ هنا استدعيت المتغير بشكل مباشر SetActive و بيطلب منك تفتح قوسين و تكتب القيمة false او true فقط .

شرح المتغير gameObject.AddComponent :

بالنسبة للمتغير AddComponent اذ بيضيف احد المكونات الى هذا الكائن مثلاً انت احبب اضيف الجاذبية او Rigidbody و للعلم ان Rigidbody احد المكونات ، هنا انت كيف تتعرف على جميع المكونات ؟ لاحظ الصورة :



لاحظ ان هذه هي جميع المكونات الي تقدر تستدعيها في البرمجة و لاحظ ايضاً انة من ضمنها الـ Script يعني ان الـ Script يعتبر من المكونات و العديد من المكونات فيك تراجعهم ، المهم وضيفة المتغير AddComponent هو الوصول الى هذة المكونات و تشغيلها او وضعها في الكائن ، هنا طريقة استخدام المتغير AddComponent في البرمجة ... تابع :



لاحظ اننا عرفت المتغير بشكل مباشر ، ثانياً بيطلب منك تفتح قوسين علشان تحدد اسم المكون و المتغير AddComponent بيطلب منك تكتب اسم المتغير كما هو موضح ، و بشكل مباشر راح يضيف المكون الى الكائن .

الى هنا نكون قد وصلنا الى نهاية شرح متغيري GameObject و Transform ، ان شاء الله تكونوا قد استفدتم من الشرح ☺.

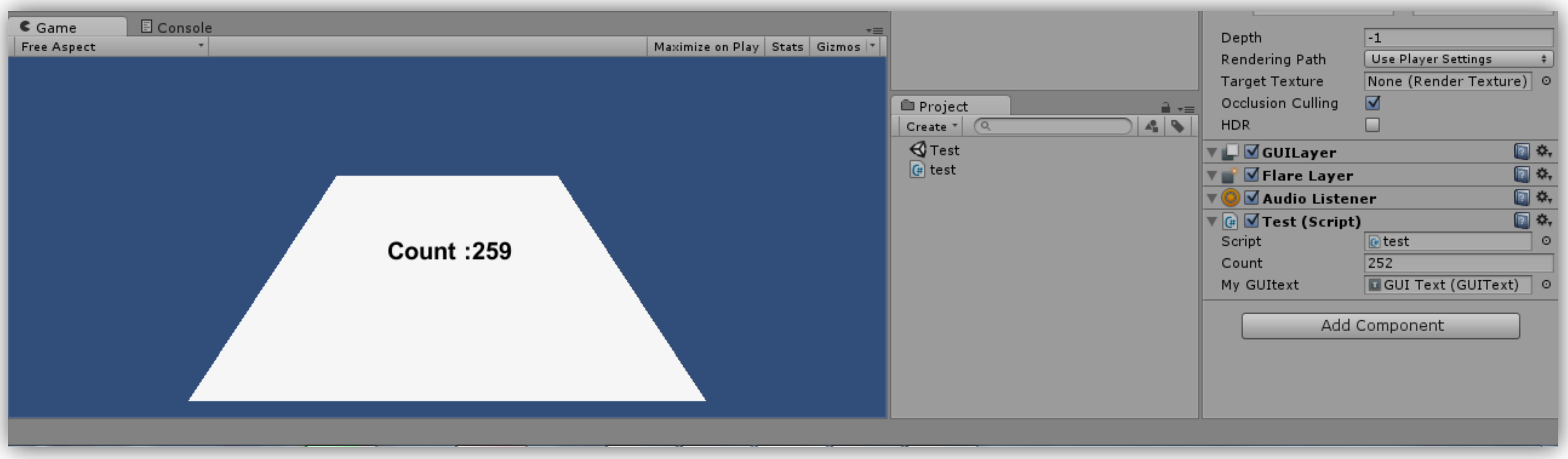
شرح GUI Text

في اليونتي GUI Text هي التي تتعامل مع الارقام و الصور في الشاشة ، بالنسبة للـ GUI Text من خلالها فيك تضيف ارقام و كتابات فقط ، لكن هنا كيف نتعامل مع الـ GUI Text في البرمجة ؟ طبعاً راح نستخدم كلاس GUIText علشان ما نطيل الشرح نفوت على البرمجة و راح تعرف المقصود .

في البداية اعمل GUI Text من Create > GUI Text طيب هنا اعمل ملف برمجي و ضعة في اي كائن غير GUI Text ، الان افتح الملف و اكتب الكود التالي :

```
test.cs
test ▸ Update ()
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6      public float Count;
7
8      public GUIText myGUItext;
9
10     void Update()
11     {
12         Count ++;
13         myGUItext.text = "Count :" + Count.ToString ();
14     }
15 }
16
```

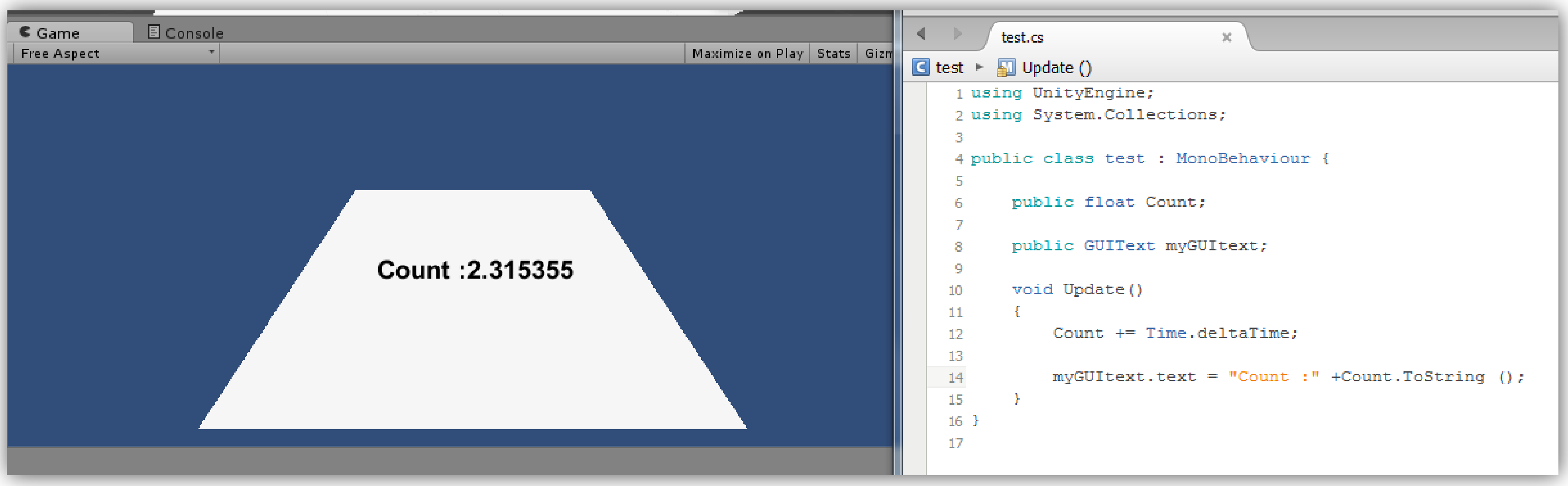
لاحظ في السطر ٨ عرفت متغير من نوع GUI Text بأسم myGUItext في Update طبعاً بعد ماكتبت متغير عددي يضيف ارقام ادا اشتغل البرنامج استدعيت متغيري myGUItext في السطر ١٣ و لانة عبارة عن text جعلتة يورث هذه القيم بصيغة text الان لما راح افتح القوسين راح يطلب مني String او نص عادي علشان يظهر في البرنامج فيك تكتب اي شيئ ، بعدها تعمل علامة + و علامة + تعني انة بجانب كلمة Count راح تضيف شيئ معين و ليكن المتغير العددي Count كما هو موضح ، ToString راح نتطرق الى شرحها لاحقاً ، الان في البرنامج اسحب الكائن GUI Text الى المتغير تبعة و قم بتشغيل البرنامج .



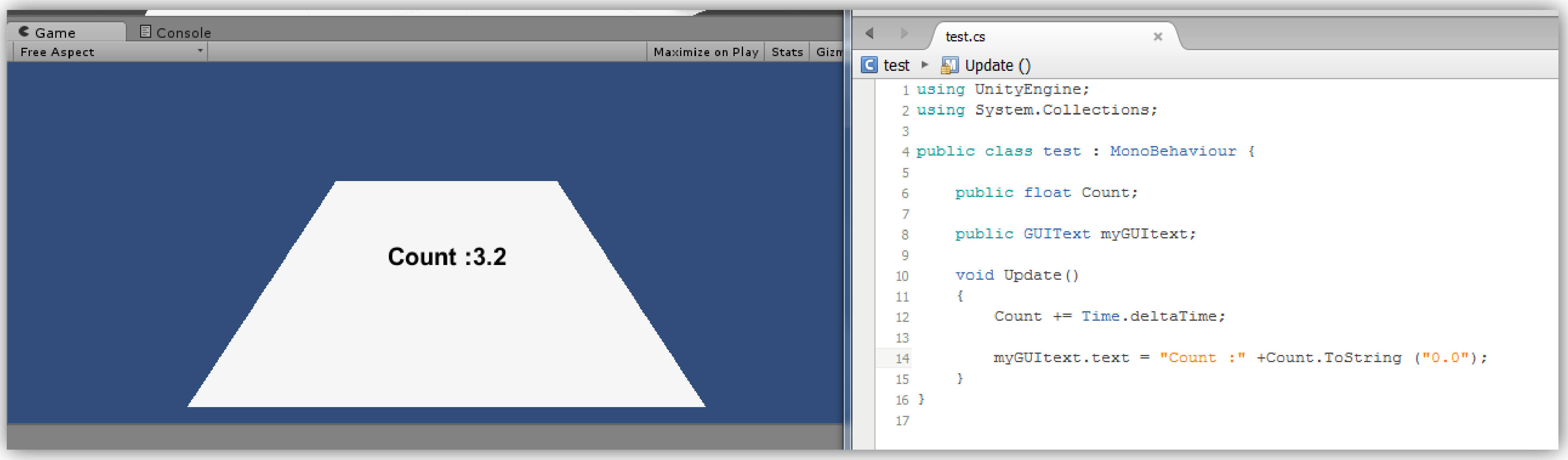
لاحظ ان GUI Text تتعامل مع الارقام و الحروف فقط ، بالنسبة لنظام ToString الي يتلاعب بطريقة ظهور الارقام ، مثلاً انت تريد هذه الارقام تتصاعد على صيغة كسور عشرية ، لاحظ هنا :

```
6      public float Count;
7
8      public GUIText myGUItext;
9
10     void Update()
11     {
12         Count ++;
13         myGUItext.text = "Count :" + Count.ToString ("0.0");
14     }
15 }
16
```

لاحظ في اسطر ١٣ في النظام ToString حددت ان هذا الرقم راح يظهر على صيغة كسر 0.0 ، هنا نظام ToString اكثر الاوقات يستخدم تتعامل مع ارقام المتغير Time لانه يظهر بأرقام كبيرة ، افترضنا ان المتغير Count يتصاعد او يزيد بنسبة المتغير Time.deltaTime هنا راح تضطر تستخدم النظام ToString علشان تحدد قيمة معينة او ارقام معينة علشان يزيد فيها ، لاحظ الكود التالي :



لاحظ في السطر ١٢ جعلت زيادة المتغير Count بنسبة ارقام Time.deltaTime ، لان لاحظ في البرنامج كيف تزايدت الارقام ، هنا علشان انت تستخدم ارقام صغيرة يعني يكفي رقمين في البرنامج بتستخدم نظام ToString و راح تحدد عدد الرقام الظاهره :

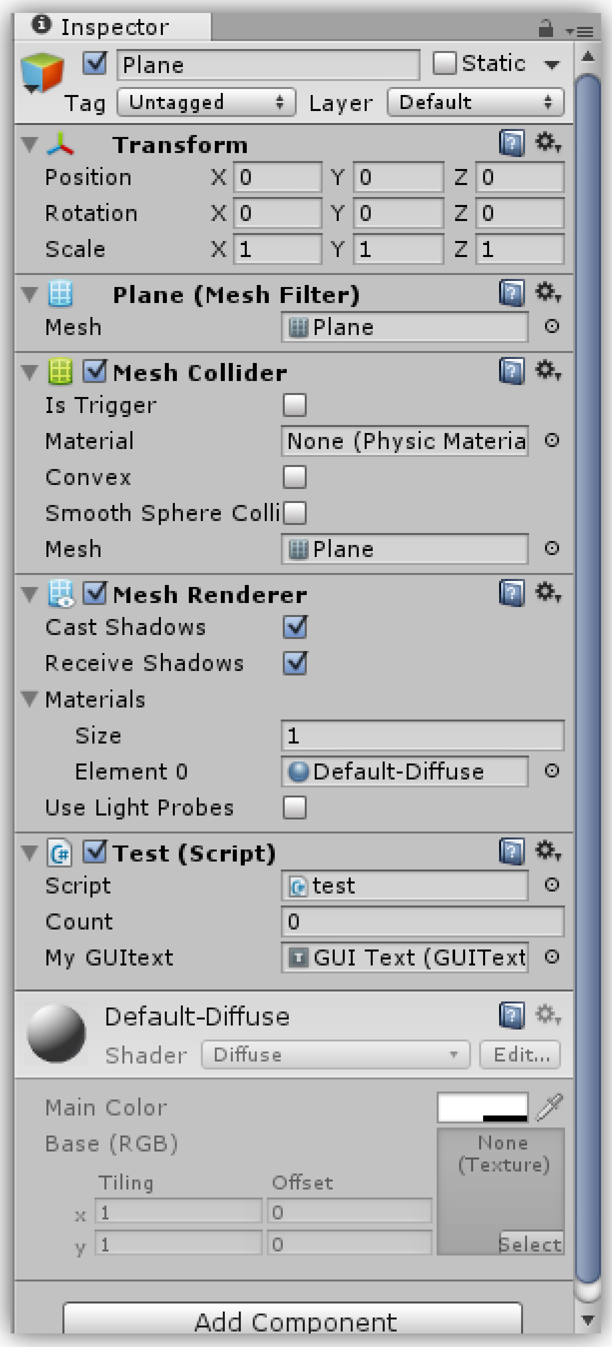


<u>شرح المتغير GUIText.color :</u>

المتغير GUIText.color متغير سهل جداً و مايحتاج شرح مطول ، وضيفتة هي انة يقوم بتغيير لون النص الى اللون الذي تريدة لاحظ الكود التالي :

```
test.cs
test ▸ Update ()
1 using UnityEngine;
2 using System.Collections;
3
4 public class test : MonoBehaviour {
5
6     public float Count;
7
8     public GUIText myGUItext;
9
10     void Update()
11     {
12         Count += Time.deltaTime;
13
14         myGUItext.text = "Count :" +Count.ToString ("0.0");
15         myGUItext.color = Color.red;
16     }
17 }
18
```

لاحظ في السطر ١٥ استخدم المتغير تبعي و استدعيت المتغير color و منة حددت لون اللنص بهذة البساطة ، انا شرحتة علشان تأخذ فكرة عن طريقة استدعاء لون الى هذا النص ، بالنسبة لكائن معين يعني انت تريد تحديد لون معين لهذا الكائن من البرمجة ؟؟ انت من فكرتك راح تستخدم gameObject.color = Color.red لكن هذه الطريقة خاطئة ، بالنسبة لجيم اوجكت ما راح تقدر تستدعي لون بهذه الطريقة ، لماذا ؟؟؟ لاحظ الصورة :



لاحظ الصورة ، هذه الصورة مأخوذة من احد الكائنات في البرنامج ، و توجد له عدة مكونات الي هي : Transform , Plane (Mesh Filter), Mesh Renderer و اخيراً ملف الاسكربت ايضاً يعد من احد المكونات، هنا انت فيك تتضيف لون هذا الكائن من Mesh Renderer > Materials و منة فيك تحدد اللون الذي تريدة ، لاحظ هنا ان علبة الالوان موجودة في الريندرر في الماتريال ، و انت علشان توصل الى علبة الالوان في البرمجة راح تضطر انك تدخل الى هذه الامتدادات و تضيف او تستخرج اللون الذي تريدة :

```
test.cs
test ▸ Update ()
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6      public float Count;
7
8      public GUIText myGUItext;
9
10     void Update()
11     {
12         Count += Time.deltaTime;
13         myGUItext.text = "Count :" +Count.ToString ("0.0");
14
15         gameObject.renderer.material.color = Color.red;
16     }
17 }
18
```

شرح المتغير GUItext.enabled :

بالنسبة للتحكم بأظهار الكائن GUI Text راح تستخدم متغير enabled و هذا المتغير من نوع bool يخزن فية قيم false , true مثل المتغير SetActive ، هنا بشكل مباشر راح تستخدم المتغير و تعين القيمة التي تريدها :

```
test.cs
test ▸ Update ()
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6      public float Count;
7
8      public GUIText myGUItext;
9
10     void Update()
11     {
12         Count += Time.deltaTime;
13         myGUItext.text = "Count :" +Count.ToString ("0.0");
14
15         myGUItext.enabled = false;
16
17     }
18 }
19
```

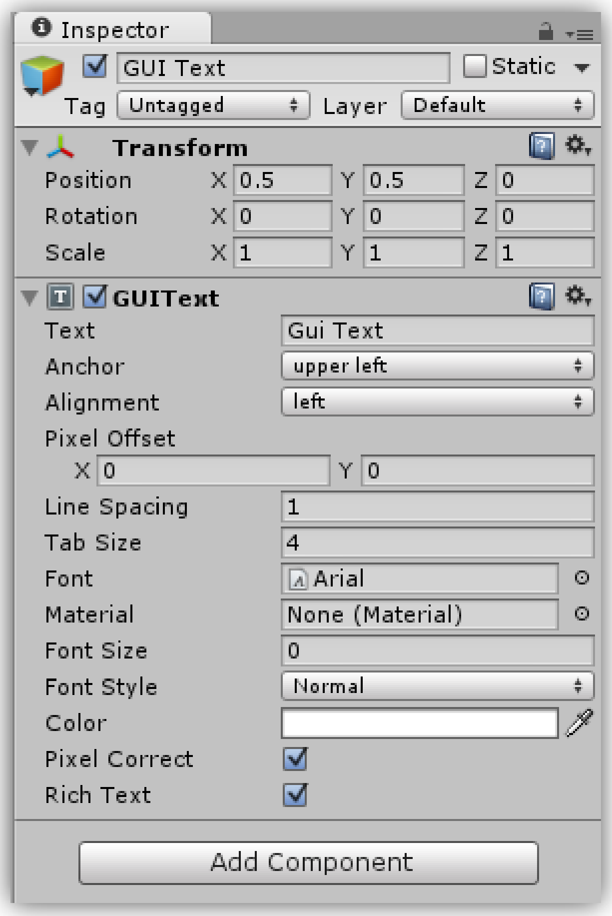
لاحظ السطر ١٥ فية معنى الكلام الي شرحتة ، هنا فيك تكتب نفس القيمة false لكن بنفس الطريقة بأستخدام علامة التعجب ! لكن بالطريقة دي انت راح تعرف القيم true , false معاً يعني راح يتضل يختفي و يظهر ، علشان نوضح المتغير enabled أكثر نستخدمة بطريقة ثانية ، راح نستخدمة في كائن الضوء Light .

قبل ما تكتب اي شيئ تأكد من ان الملف موضوع داخل كائن فية ضوء Directional light او اي ضوء آخر ، الان ارجع الى الملف و اكتب الكود التالي :

```
test.cs
test ▸ Update ()
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6      public float Count;
7
8      public GUIText myGUItext;
9
10     void Update()
11     {
12         Count += Time.deltaTime;
13         myGUItext.text = "Count :" +Count.ToString ("0.0");
14
15         if(Input.GetKeyDown ("space"))
16         {
17             light.enabled = !light.enabled;
18         }
19
20     }
21 }
22
```

لاحظ من السطر ١٥ الى السطر ١٨ ، في الشطر كتبت انة في حالة ضغطنا على المفتاح space راح يشتغل الكود التالي ، الكود وضيفتة راح يشغل الضوء و في الضغطة الثانية راح يطفية و هكذا و هذا شرح المتغير enabled .

GUI Text يحتوي على العديد من المتغيرات و فيك تستخدمها حسب رغبتك ، اساس المتغيرات هو الكائن نفسة اي ان المتغيرات مأخوذة من خصائص الكائن نفسة :



هنا مثلاً تريد تستخدم الـ text في البرمجة راح تستخدم guitext.text بهذا الشكل ، جميع المتغيرات تبع الكلاس GUI Text موجودة في الكائن بش بشكل عام مثلاً انا اريد اتعامل مع الخيار Anchor الي راح يضبط موقع GUI Text في البرنامج نرجع الى البرمجة و نتعامل معه بهذا الشكل :

```
void Update()
{
    guiText.anchor = TextAnchor.MiddleCenter;
    guiText.alignment = TextAlignment.Center;
}
}
```

كما هو موضح بشكل مباشر فيك تعرف المتغير و تختار اي خيار تريدة في الكائن في البرمجة ...

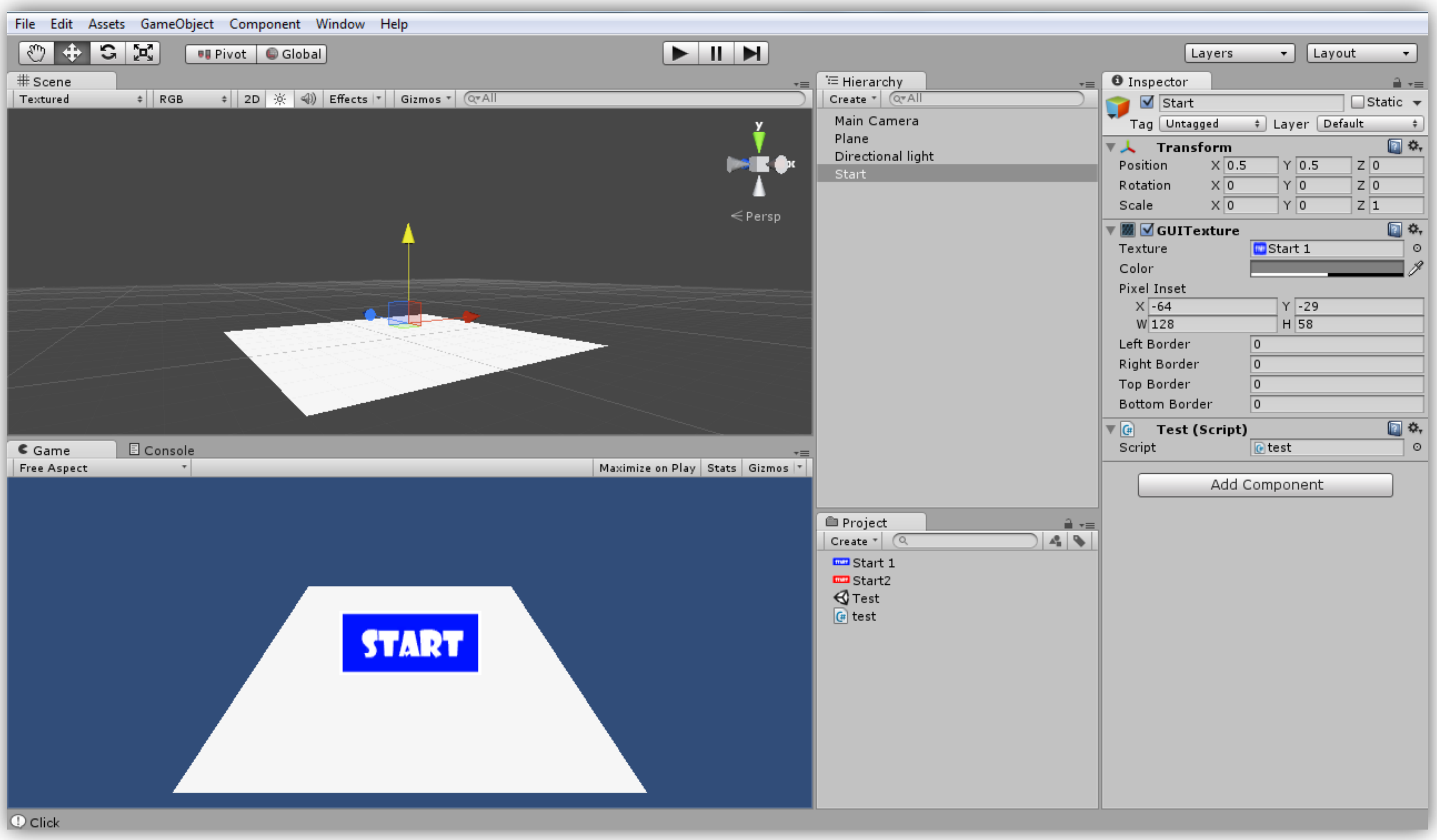
شرح متغير GUITexture

الكائن GUITexture هو الي يتعمل مع الي Texture2D في البرنامج ، مثلاً انت عندك صورة ولتكن هذة الصورة 

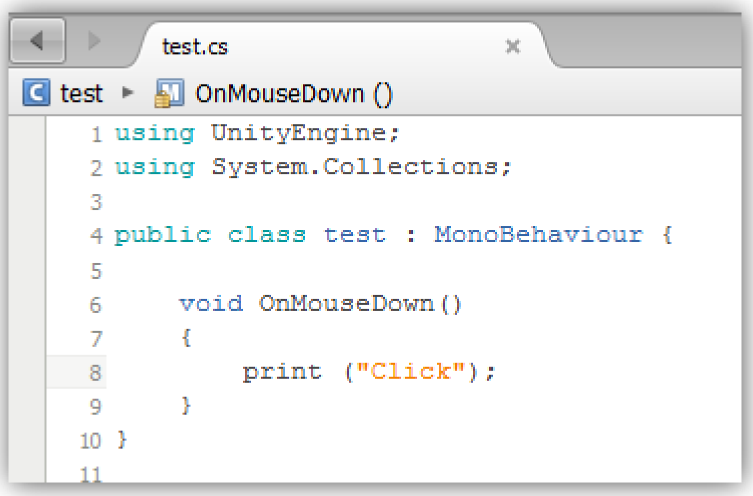
في هذه الحالة فيك تسحب الصورة الى الـ GUITexture ، ايضاً فيك تتعامل مع GUITexture على اساس انة مفتاح او زر او اي شيئ في البرمجة مافي شيئ مستحيل ،بعيداً عن الحديث نحاول نصنع شيئ من GUITexture علشان نعرف كيف يستخدم :

اولاً انشئ GUITexture و قم بسحب اي صورة الية .

ثانياً انشئ ملف برمجي و راح اشرح تفاصيل الملف قبل الكتابة : الملف راح يكون عبارة عن ملف تعريفي راح يجعل هذا الـ GUITexture يأخذ خواص الزر ، هنا راح نستخدم ذالة من اصل ٤ دوال و التي هي OnMouseDown ، هذه الذالة هي الي راح تنشأ حدث في حالة قمت بالضغط على هذا الكائن ، اولاً قم بأنشاء GUITexture كما هو موضح :



لاحظ موضعة في الصورة ، الان قم بأنشاء ملف برمجي و اكتب الكود التالي :



هذا الكود راح يعرف لنا شيئ انه ادا ضغطنا الى الكائن راح يطبع لنا Click ، لاحظ هنا استخدمنا دالة خاصة بتحكمات الماوس بالنسبة للضغط يعني في حالة ضغطت على هذا الكائن راح يطبع هذا الكلام ، طيب ما علاقة هذه الدالة بالمتغير GUITexture ؟؟؟ انا استخدمت هذة الدالة من اصل ٤ دوال و الي هم Down , Exit , Enter , Up وهذه الاربع الدوال ،

هنا راح نعمل شيئ جديد باستخدام دالة ثانية Enter لاحظ هنا

اولاً استخدم صورة ثانية بجانب الصورة الاولى 

بحيث ان الصورة الاولى راح تكون زرقاء و الصورة الثانية راح تكون حمراء ومتى راح تكون حمراء ؟؟ راح تكون حمراء ادا مر الماوس على الصورة ،تابع ...

اكتب الكود التالي في نفس الملف :

test.cs

test ▸ OnMouseDown ()

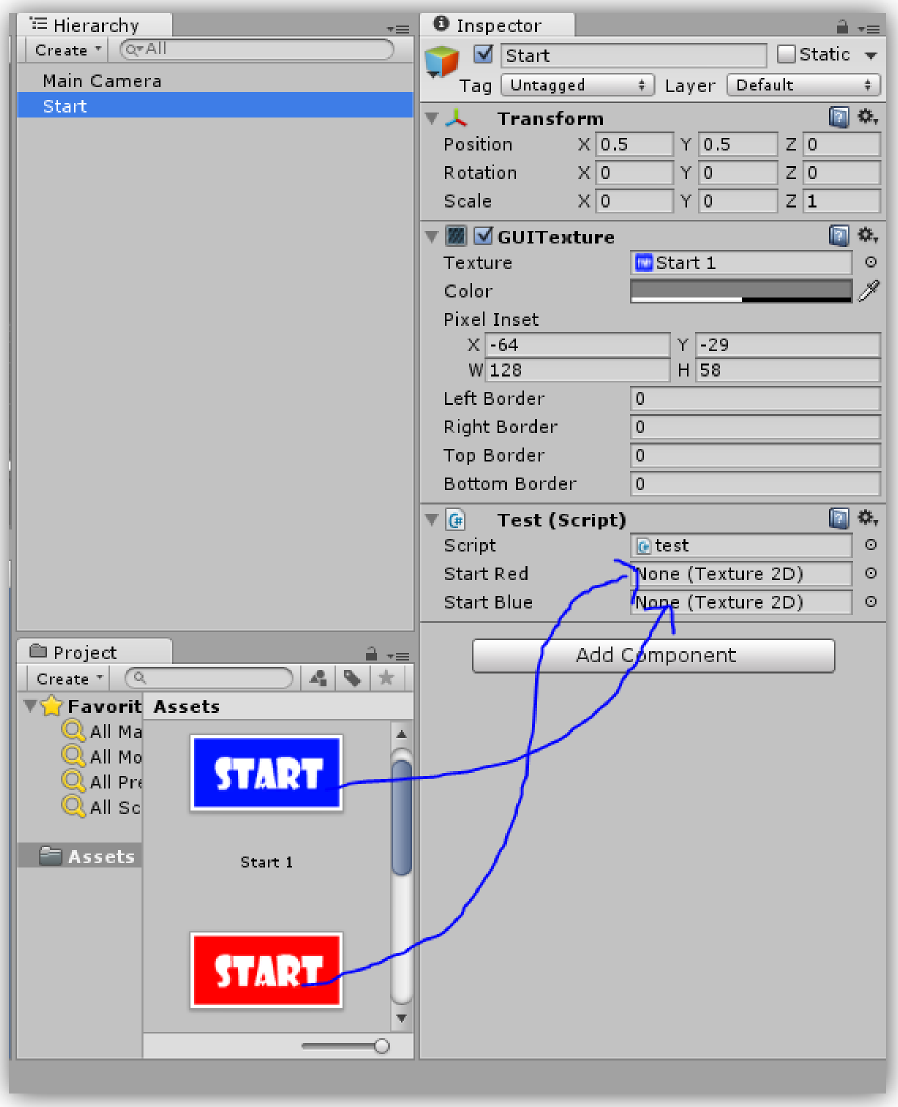
```
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6      public Texture2D StartRed;
7      public Texture2D StartBlue;
8
9      void OnMouseEnter()
10     {
11         guiTexture.texture = StartRed;
12     }
13     void OnMouseExit()
14     {
15         guiTexture.texture = StartBlue;
16     }
17     void OnMouseDown()
18     {
19         print ("Click");
20     }
21     void OnMouseUp()
22     {
23         print ("");
24     }
25 }
26
```

لاحظ في البداية انني استخدمت ٤ دوال الي هم Enter : و دي الدالة وضيفتها انها تقرأ اي شيئ يمر في هذا الكائن او يدخل فية اما الدالة Exit برضوا تقرأ اي شيئ يخرج من هذا الكائن .

الدالة Up , Down خاصتان بالضغط يعني في حالة بدأت بالضغط راح يحصل شيئ معين هذا بالنسبة للدالة Down اما الدالة Up هي في حالة انك خرجت من الضغط او تركت الضغط هذا المعنى ، في النهاية هم دوال خاصين بتحكمات الماوس OnMouse يعني انه اذا حصل شيئ معين في الماوس، و اعطاك ٤ دوال فيك تستخدمهم بأكثر من طريقة و لا يقتصر الامر على متغيرات GUI انما استخدمتها في GUI علشان ابسط الشرح .

لاحظ في السطرين 6,7 عرفت متغيرين من نوع Texture2D لانة يوجد معي صورتين او بالاصح تكسترين الاول الي هو Start الازرق و الثاني الاحمر، المقصود من المتغيرين راح اجعلهم يتبادلوا الادوار يعني في حالة مر الماوس على الزر راح يتغير التكستر تبعة الى StartRed و اذا خرج راح يرجع الى لونة الطبيعي StartBule، هنا انت علشان تخزن المتغيرين في GUITexture راح تستخدم متغير guiTexture وهذا المتغير الموجود في Component يعني احد المكونات في GUITexture ، في السطر ١١ في حالة مر الماوس على التكستر او دخل فية راح يتغير لونة الى الاستارت الاحمر و اذا خرج منة كما في السطر ١٥ راح يرجع لونة الاصلي، الدالة OnMouseDown راح تطبع لنا Click في حالة ضغطنا على الزر و الدالة Up ما راح تطبع شيئ اذا تركنا الضغط الى هنا اكون شرحتة بكل تفصيل ، الان نرجع الى البرنامج علشان نسحب التكستر الى مكانها لاحظ الصورة :

الان ما عليك الا انك تقوم بتشغيل البرنامج وشوف النتيجة ...

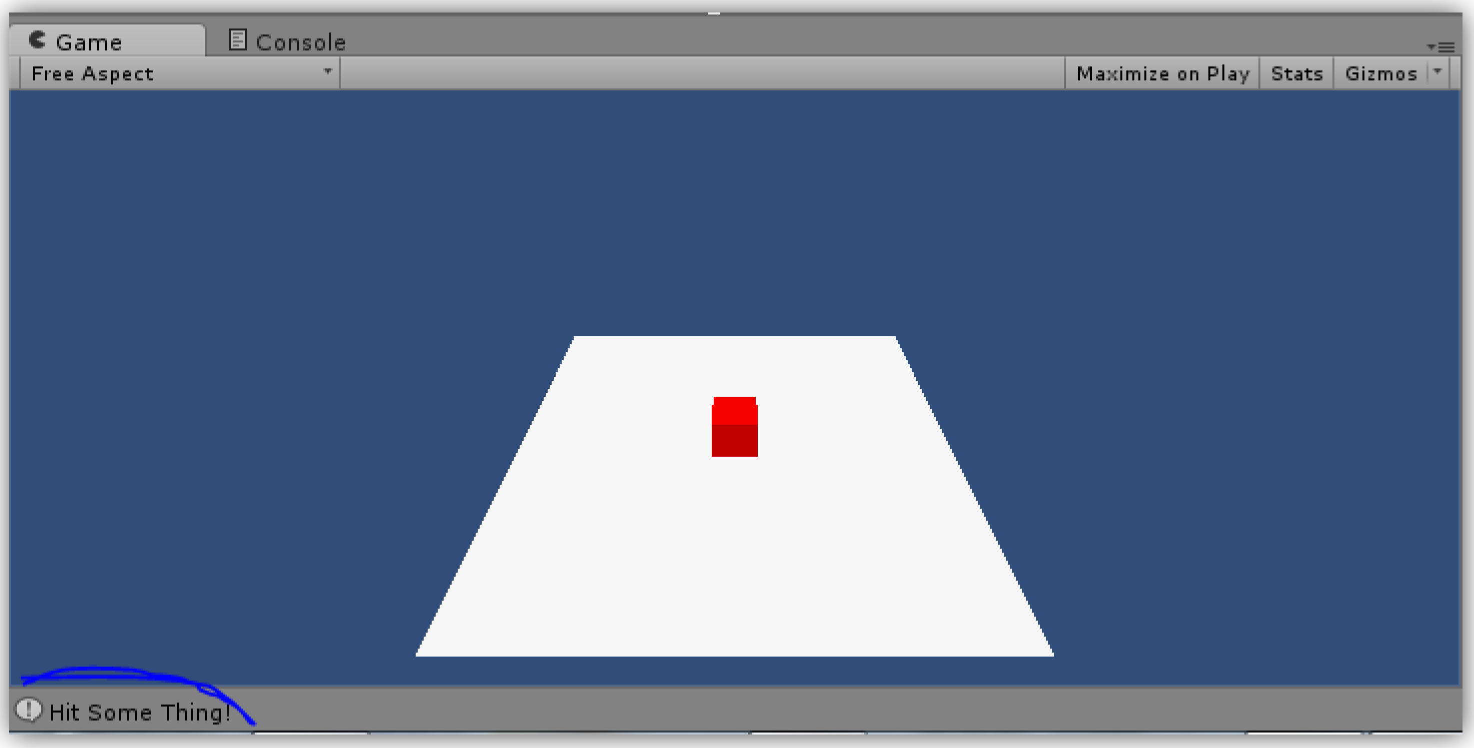


شرح دالة OnCollision

لاحظت في الدروس السابقة كنا نتحدث عن المتغيرات و كيف تتعامل معها في الدرس الاخير شرحنا بعض الدوال الخاصة بالماوس ، بالنسبة لهذا الدرس راح نشرح دالة مهمة جداً وراح نتعرف على طريقة استخدامها و هي دالة مهمة جداً قد تساعدك كثير ، دالة OnCollision وضيفتها هي انها راح تعمل حدث معين اذا حصل ان جسمين تصادموا ، طيب الدالة لها نوعان من التصادم ، الاول الي هي دالة OnCollisionEnter عندما يحصل التصادم (عندما يتصادم الجسمان) ، و الدالة الثانية OnCollisionExit عندما يخرجا من التصادم (عندما ينفصل الجسمان عن بعضهما) بمعنى آخر الدالتان تعنيان الكائن الداخلة في التصادم و الخارجة منه ، بالنسبة لطريقة استخدام الدالة راح نستخدم كلاس Collision و هذا الكلاس منة نوع ، واحد خاص بألـ Collider3D و الآخر خاص بالـ Collider2D طبعاً الخاص بالـ 3D ينكتب بهذا الشكلCollision اما بالنسبة للـ2D ينكتب بهذا الشكل Collision2D و دا يستخدم في الالعاب الـ 2D ، الى الان نكتفي بالشرح و راح نتطرق الى البرمجة و كيف نتعامل مع الدالة ... لاحظ الكود التالي:

```
test.cs
test ▸ OnCollisionEnter (Collision Col)
1  using UnityEngine;
2  using System.Collections;
3
4  public class test : MonoBehaviour {
5
6
7      void OnCollisionEnter(Collision Col)
8      {
9          if(Col.gameObject.collider)
10         {
11             print ("Hit Some Thing!");
12         }
13     }
14 }
15
```

لاحظ هنا استخدمت الدالة بشكل مباشر ، في الدالة راح يطلب منك تحدد نوع التصادم لهذا الكائن ، و Col تعني الكائن الآخر الذي راح نعمل معة التصادم لاحظ قبل كلمة Col استخدمت كلاس Collision اد الي بيحدد نوع التصادم لهذا الكائن و الي هو عبارة عن تصادم عادي ، بالنسبة للشرط راح تحدد ان هذا الكائن الي اسمة Col راح يكون اي كائن فية كولايدر لاحظ انني لما احدد كائن معين او اكتب اسم كائن معين في السطر ٩ حددت بشكل مباشر اي كائن فية كولايدر راح يكون بأسم Col و راح تتخزن فية خاصة التصادم مع الكائنات Collision و علشان تعرف ان الشرط اشتغل اعمل طبع لكلمة معينة تظهر لك ان الكائن اصطدم ...لاحظ الصورة :



لاحظ انة طبع لنا كلمة "Hit Some Thing" عندما صدم في الكائن الاخر ، طيب هنا راح نستخدم الدالة الثانية الي راح تسمح كلمة "Hit Some Thing" اذا خرج الكائن عن التصادم OnCollisionExit بشكل مباشر نستخدمها :

test.cs — test ▸ OnCollisionExit (Collision Col)

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class test : MonoBehaviour {
5
6
7     void OnCollisionEnter(Collision Col)
8     {
9         if(Col.gameObject.collider)
10        {
11            print ("Hit Some Thing!");
12        }
13    }
14
15    void OnCollisionExit(Collision Col)
16    {
17        if(Col.gameObject.collider)
18        {
19            print ("");
20        }
21    }
22 }
23
```

لاحظ من السطر ١٥ الى ٢١ استخدمت دالة OnCollisionExit نفس الشيئ راح نحدد نوع التصادم و اسمة ، و بالنسبة للطبع فيك تخلي الكلام فاضي ، هنا راح تلاحظ ان الدالتان لهما نفس طريقة التعريف و الكتابة مافي اي شيئ صعب فيهما ، مثال راح نجعل هذا الكائن يصطدم في كائن آخر و يدمره ، نستخدم نفس الدالة لاحظ :

test.cs — test ▸ OnCollisionEnter (Collision Col)

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class test : MonoBehaviour {
5
6
7     void OnCollisionEnter(Collision Col)
8     {
9         if(Col.gameObject.collider)
10        {
11            Destroy (Col.gameObject);
12        }
13    }
14 }
15
```

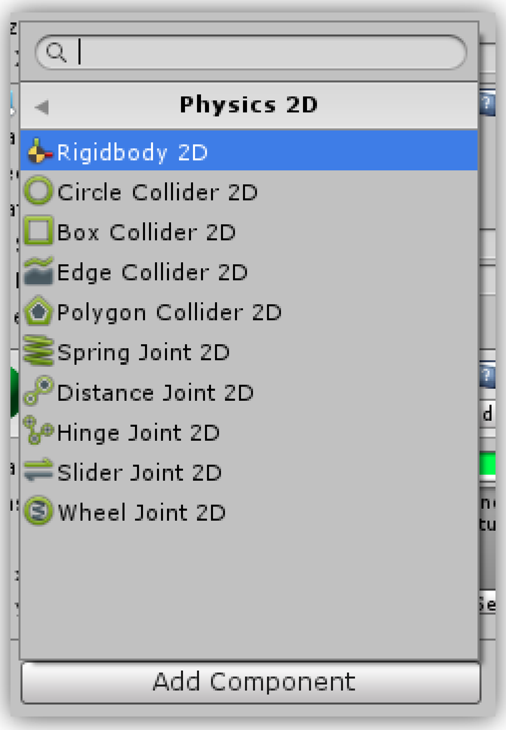
لاحظ في السطر ١١ ان الكائن اذا اصطدم في اي كائن راح يدمره بس لاحظ في Destroy راح نستخدم Col علشان تحدد ان التدمير راح يكون في الكائن الآخر .

الان اجعل كائنان يتصادمان و لاحظ ان الآخر سوف يختفي ....!

الى هنا اكون قد شرحت دالة OnCollision بشكل كامل حاول تستخدم الدالة في اي شيئ و راح تعرف كيف تتعامل معها و راح تساعدك كثير ، اما بالنسبة لدالة

OnCollisionEnter2D , OnCollisionExit2D

هنا تكتبهم بهذا الشكل و في القوسين تستخدم Collision2D و الاسم اي اسم اختار ، هنا الدالة 2D راح تعمل اذا كان الكائن مكون من Collider2D في تجده من Physics2D :



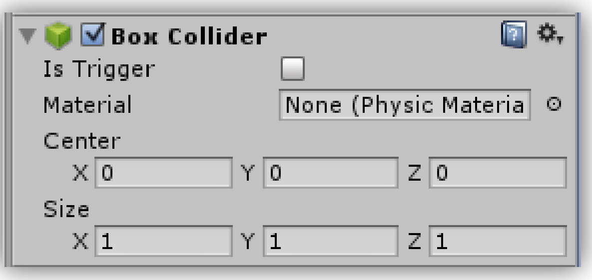
وفيك تستخدم أي كولايدر 2D و تتعامل معه في الدالة بشكل مباشر ، اما بالنسبة لتحديد كائن معين يحصل التصادم فية هو عن طريق متغير name او tag لاحظ

```
7     void OnCollisionEnter(Collision Col)
8     {
9         if(Col.gameObject.name == "Plane")
10        {
11            print ("Hit The Plane");
12        }
13    }
14 }
```

اما tag غير كلمة name الى tag و اكتب اسم التاج و راح يشتغل بشكل جيد .

شرح دالة OnTrigger

دالة OnTrigger شبيهه بدالة OnCollision و طبعاً دالة OnTrigger لها نوعين عندما يدخل كائن في كائن و IsTrigger مفعل و الثاني عندما يخرج من الكائن و IsTrigger مفعل ، ايش يعني هذا الكلام ، لاحظ اننا في الشرح السابق تركت الكولايدر كما هو لم اقم بتغيير اعداداتة ، لاحظ الصورة التالي :



هذه الصورة مأخوذة من احد الكائنات لا يهم ، المهم هو ان نضرك يتركز على زر Is Trigger لاحظ ان هذا الزر يتحكم بتفعيل الكولايدر او لا ، اذا فعلت Is Trigger و قمت بجعل كائن لآخر يصطدم فية راح تلاحظ انة يخترقة اي كأنه الكولايدر ليس له وجود ابدا بس يكون له عمل معية في الدالة

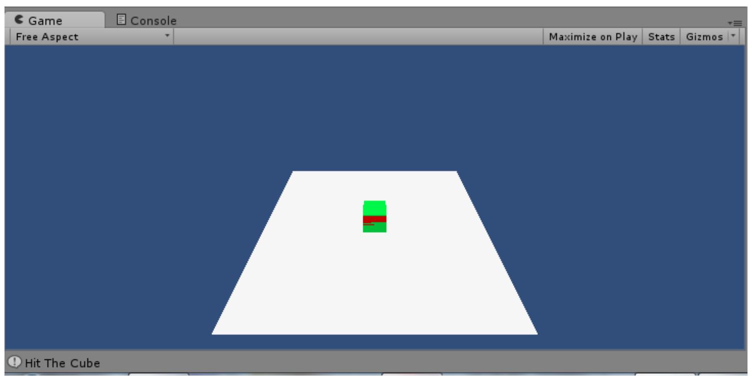
في البداية اعمل كائنين الاول فية Is Trigger مفعل و الاخر لا ، هنا اعمل ملف برمجي و ضعة في الكائن الذي فية Is Trigger غير مفعل ... الان لاحظ هذا الكود باستخدام دالة OnTriggerEnter :

```
test.cs
test ▸ OnTriggerEnter (Collider Col)
using UnityEngine;
using System.Collections;

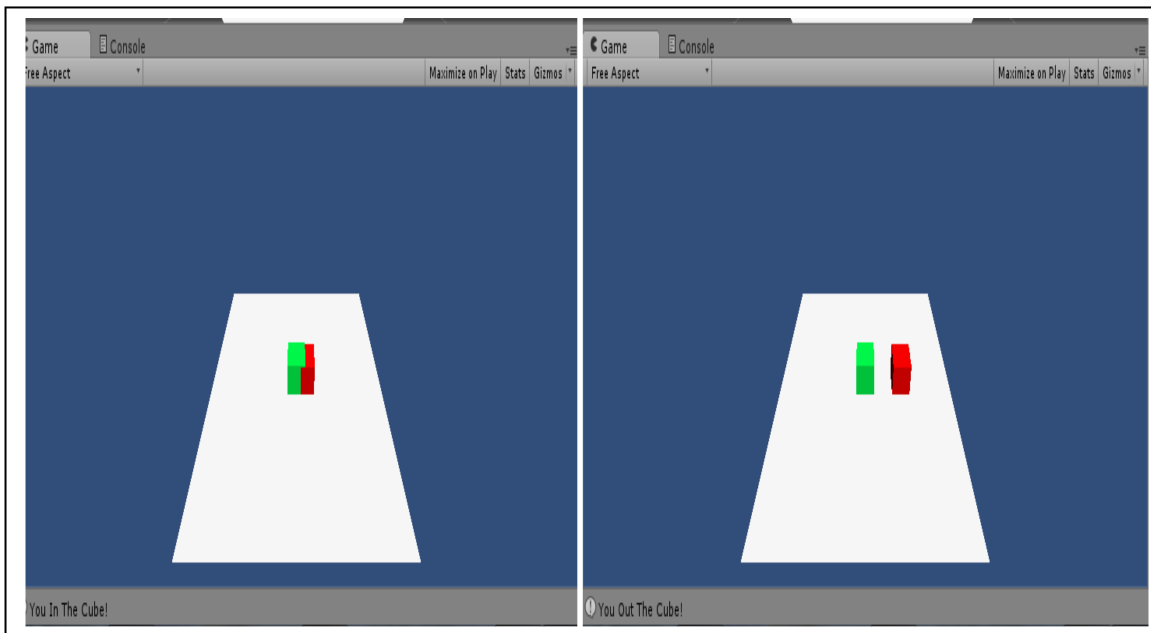
public class test : MonoBehaviour {


    void OnTriggerEnter(Collider Col)
    {
        if(Col.gameObject.name == "Cube")
        {
            print ("Hit The Cube");
        }
    }
}
```

لاحظ هنا اننا استخدمنا دالة OnTriggerEnter نفس الشيئ لكن في القوسين حددت ان الكائن راح تعامل مع الكائن الاخر بصفة Collider كما هو موضح باقي الكود ضعة كما هو الان لاحظ الصورة التالي من داخل العملية :



لاحظ انه عندما دخل الكائن الاول في الثاني ظهرت كلمة "Hit The Cube" هنا فيك تتعامل مع Destroy علشان تتحكم بظهور و اختفاء الكائن ، لاحظ الكود التالي مع الحدث طبعاً مع استخدام دالة OnTriggerExit :




```
test.cs
test ▸ OnTriggerExit (Collider Col)
using UnityEngine;
using System.Collections;

public class test : MonoBehaviour {


    void OnTriggerEnter(Collider Col)
    {
        if(Col.gameObject.name == "Cube")
        {
            print ("You In The Cube!");
        }
    }

    void OnTriggerExit(Collider Col)
    {
        if(Col.gameObject.name == "Cube")
        {
            print("You Out The Cube!");
        }
    }
}
```

لاحظ الكود و طريقة عملة طبعاً الدالة شبيهه بالدلة OnCollision ولهم نفس العمل لكن بطريقة مختلفة ، هنا علشان تتعامل مع مكونات الكائن هنا راح تقدر تستخدم Destroyعلشان تحذف مكون معين ماعدا الكولايدر ، لاحظ :

```
using UnityEngine;
using System.Collections;

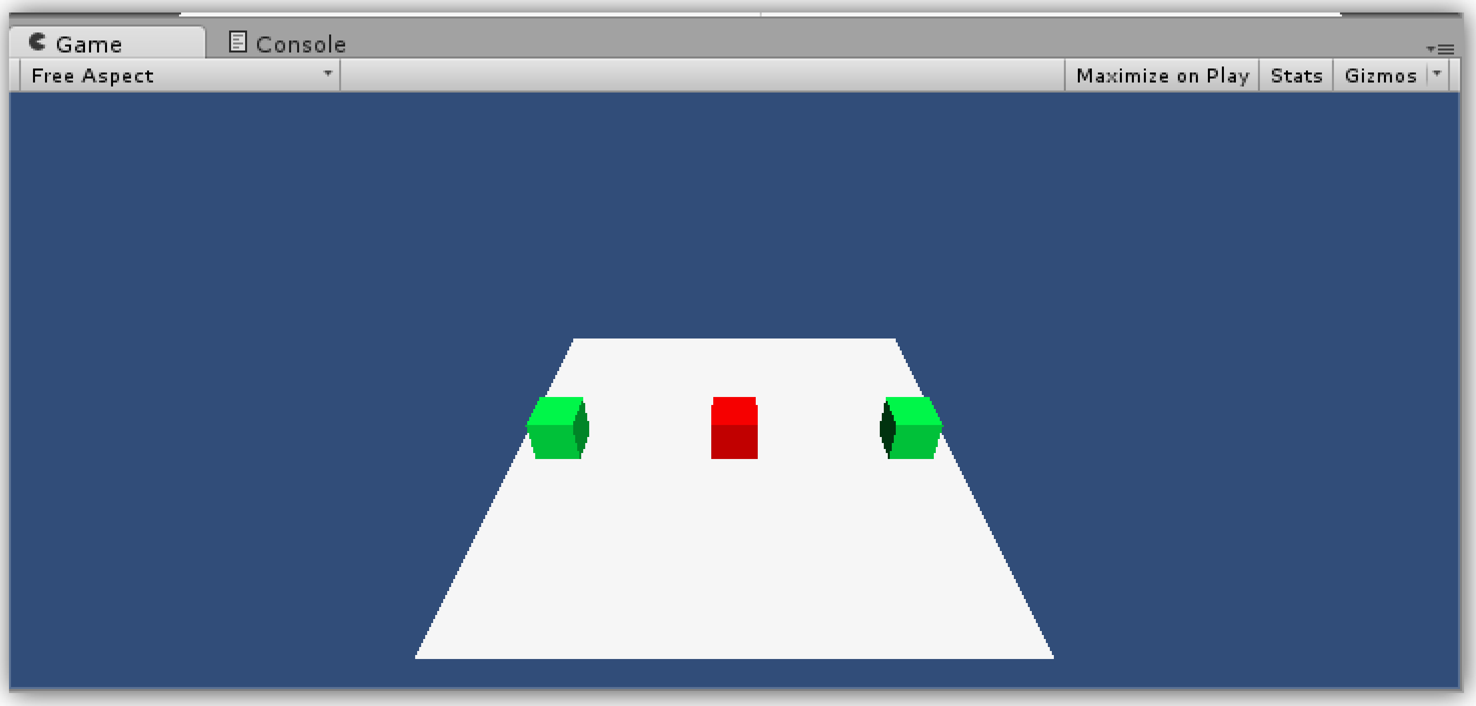
public class test : MonoBehaviour {


    void OnTriggerEnter(Collider Col)
    {
        if(Col.gameObject.name == "Cube")
        {
            Destroy (Col.gameObject.GetComponent<MeshRenderer>());
        }
    }

    void OnTriggerExit(Collider Col)
    {
        if(Col.gameObject.name == "Cube")
        {
            Col.gameObject.AddComponent<MeshRenderer>();
        }
    }
}
```

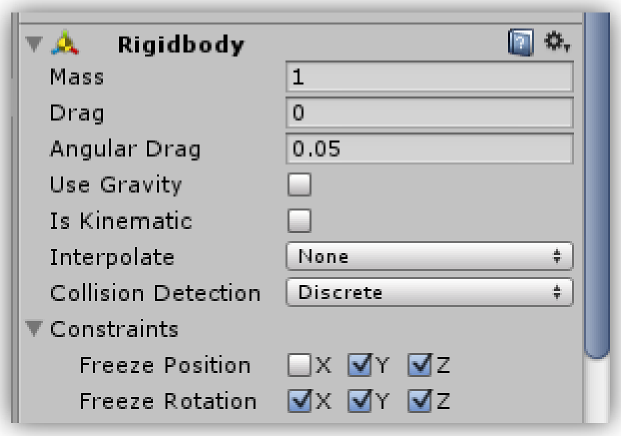
طبعاً فيك تستخدم جميع المتغيرات في جميع الدوال و كلاً حسب فكرتة ، راح نعمل شيئ بسيط ،راح نجعل الكائن تبعنا يتحرك الى اليمين و اذا صدم في كائن معين راح يتحرك الى اليسار و اذا صدم في كائن آخر راح يتحرك الى اليمين لاحظ هنا الكائنين راح نجعلهم مخفيين بحيث انه راح يظهر لنا الكائن تبعنا يتحرك الى اليسار و الى اليمين بشكل تلقائي ، تابع الخطوات التالية :

انشأ مكعبين بحيث يكون موقع الاول في position على محور x يساوي 4 و الآخر يساوي -4 و الان انشا مكعب آخر و اجعل موقعة في الوسط كما هو في الشكل التالي :



الان المكعبين باللون الاخضر اعمل لهم تاج بأسم Finish .

ثانياً : المكعب باللون الاحمر اعمل فية Rigidbody و اضبط اعداداته كما هو في الشكل :



تأكد من ان Is Trigger غير مفعل في كل المكعبات .

ثالثاً : اعمل ملف برمجي جديد و قم بكتابة الكود التالي :

test.cs

test ▸ Update ()

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class test : MonoBehaviour {
5
6     public float Dir = 1;
7     public float Speed = 10;
8
9     void Update()
10     {
11         transform.Translate (new Vector3 (1, 0, 0) * Time.deltaTime*Speed*Dir);
12     }
13
14     void OnCollisionEnter(Collision Col)
15     {
16         if(Col.gameObject.tag == "Finish")
17         {
18             Dir *= -1;
19         }
20     }
21 }
22
```

الان هنا قبل ما تبدأ بتعقيد أي مسألة عليك لاحظ معي اولاً علشان تعرف كيف تفهم الأكواد ، بالنسبة للدوال لاحظ هنا اننا استخدمت دالتين الاول هي Update و الثانية هي OnCollisionEnter ، بالنسبة للمتغيرات معنا متغيرين من نوع float بحيث ان الاول هو Dir هو الذي راح يغير اتجاه حركة الكائن من اليمين الى اليسار و الثاني Speed هو الي راح يتحكم بسرعة الحركة ، في السطر ١١ لاحظ اننا عرفت حركة على محور x في الكائن ، و لاحظ اننا ضربت فارق الوقت مع السرعة و الاتجاه طبعاً هو بيتعامل مع الاتجاه على اساس انة متغير من نوع float بس لاحظ هنا اذا صارت Speed = -Speed راح يتغير اتجاه الحركة الى السالب ، بس انا علشان ما اعقد المسألة استخدمت متغير آخر علشان اغير فية الحركة ، لاحظ في السطر ١٨ في دالة التصادم راح يظل المتغير ينضرب في 1- علشان تتغير قيمتة العددية بين الاشارتين السالب و الموجب ، علشان ما تتعقد الامور لاحظ هذا القانون :

+ = - * -
+ = + * +
- = + * -
- = - * +

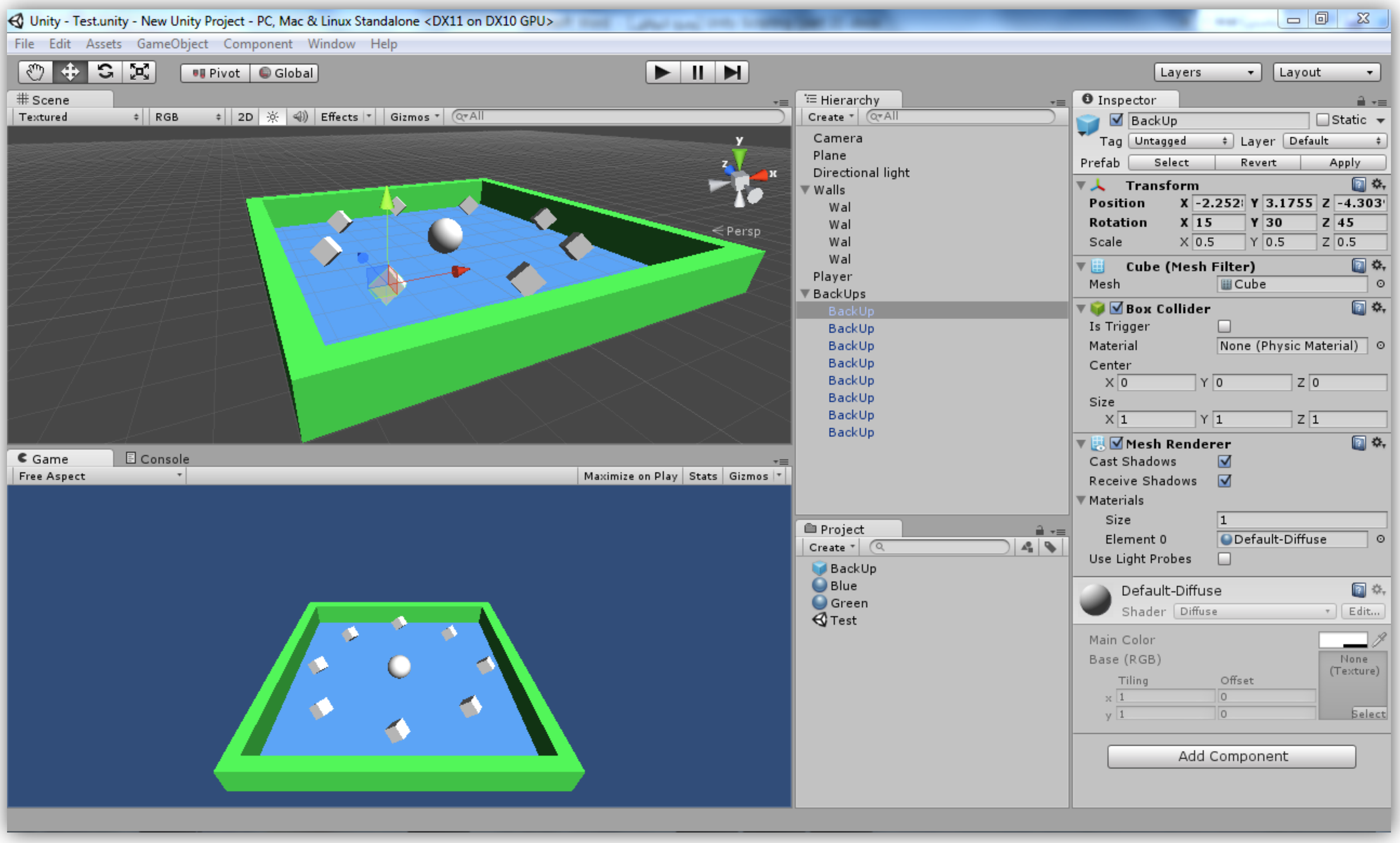
مش مشكلة اذا ما فهمت القانون في الاعلى المهم انك فهمت الملف في الاعلى ، الى هنا قم بوضع الكود في المكعب الاحمر و لاحظ طريقة عملة …

تصميم لعبة بسيطة بأسم Roll A Ball

بالنسبة للعبة Roll A Ball هي من اختراع شركة Unity3D الشركة عملت اللعبة بعد تعلم اساسيات البرنامج علشان تتيح لك التعرف على المتغيرات و الدوال و كيف تتعامل معها في تصميم لعبة بسيطة في اليونتي ،صحيح قد يكون البعض منكم مر على اللعبة الى اننا الان راح شرحها بشكل مفصل علشان تفهم كيف تتعامل مع الدوال و المتغيرات لخلق لعبة بسيطة .

في البداية التصميم راح يكون عليك ،اعمل التصميم التالي :

التالي



هنا راح نستخدم البرمجة ، في البداية لاحظ ان عدد الـ Backups هو ٨ الان اعمل ملف برمجي بأسم RotateBackups و اكتب الكود التالي فية :

هذا الكود الي راح يقوم بتدوير الـ Backups حول نفسها ، يعني ان هذا الملف راح تطرحه في جميع الـ Backups ...

```
RotateBackups.cs
RotateBackups ▸ Update ()
using UnityEngine;
using System.Collections;

public class RotateBackups : MonoBehaviour {


    void Update () {

        transform.Rotate (new Vector3 (15, 30, 45) * Time.deltaTime);
    }
}
```

هنا راح نعمل ملف جديد بأسم PlayerController في البداية راح نبرمج حركة الكرة ... تابع :

RotateBackups.cs | PlayerController.cs

PlayerController ► Update ()

```
using UnityEngine;
using System.Collections;

public class PlayerController : MonoBehaviour {

    public float Speed = 200;


    void Update () {

        float MoveH = Input.GetAxis ("Horizontal");
        float MoveV = Input.GetAxis ("Vertical");
        Vector3 movement = new Vector3 (MoveH, 0, MoveV);
        rigidbody.AddForce (movement * Speed * Time.deltaTime);
    }
}
```

طبعاً لو تلاحظ اننا استخدمنا هذا الكود سابقاً في عمل حركة للحركة ، هنا قبل ان تضع الملف في الكرة قم بعمل Rigidbody في الكرة ،هنا الملف بيحدد لنا ان الكرة راح يكون لها حركة بشكل افقي و راح تتضاف هذه الحركة في السرعة في الجاذبية علشان يخلق لنا حركة كرة واقعية، هنا راح تضيف الملف في الكرة بشكل مباشر ، لكن تأكد من وضع خاصية Rigidbody في الكرة ...

الان بعدما تحركت الكرة ،راح نبرمج اختفاء الـ Backups لكن تأكد في البداية ان زر Is Trigger في Backups مفعل ، هنا اكتب الكود التالي :

RotateBackups.cs | PlayerController.cs

PlayerController ► OnTriggerEnter (Collider Col)

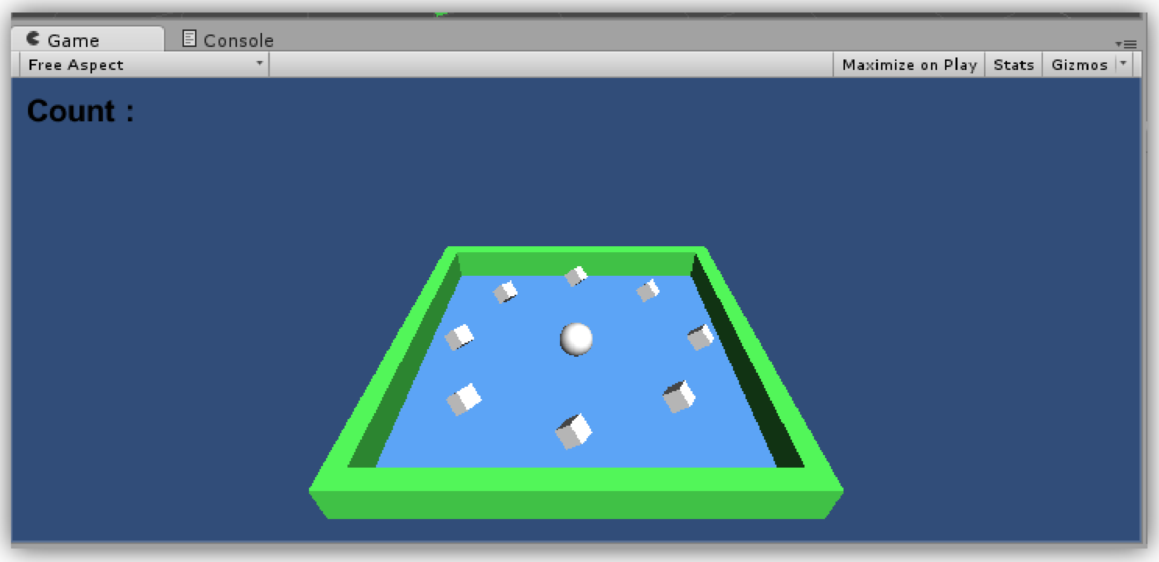
```
1  using UnityEngine;
2  using System.Collections;
3
4  public class PlayerController : MonoBehaviour {
5
6      public float Speed = 200;
7
8
9      void Update () {
10
11         float MoveH = Input.GetAxis ("Horizontal");
12         float MoveV = Input.GetAxis ("Vertical");
13         Vector3 movement = new Vector3 (MoveH, 0, MoveV);
14         rigidbody.AddForce (movement * Speed * Time.deltaTime);
15     }
16
17     void OnTriggerEnter(Collider Col)
18     {
19         if(Col.gameObject.name == "BackUp")
20         {
21             Col.gameObject.SetActive (false);
22         }
23     }
24 }
25
```

لاحظ في السطر ١٧ في نفس الملف استخدمت الدالة OnTriggerEnter علشان ما يحصل تصادم بين الكرة و الـbackups و يظهر انة في تصادم ، لكن بهذا الشكل ما راح تلاحظ انة في تصادم بل بشكل سلس راح تختفي ، الكود في الدالة مفهوم الي راح يقوم بالإخفاء ...

الان ارجع الى البرنامج و جرب عمل الكرة و اصدمها في Backup و لاحظ انة راح يختفي ،

الان راح نعمل شيئ آخر راح نعمل عدد ، مثلاً اذا صدمت الكرة في أي Backup يزيد الرقم بواحد لكن هنا راح نستخدم GUI Text علشان يظهر الرقم في الشاشة ... تابع :

اولاً قم بأنشاء GUI Text و اضبط موضعه في الشاشة كما هو موضح :



الان راح نرجع الى نفس الملف في الكرة

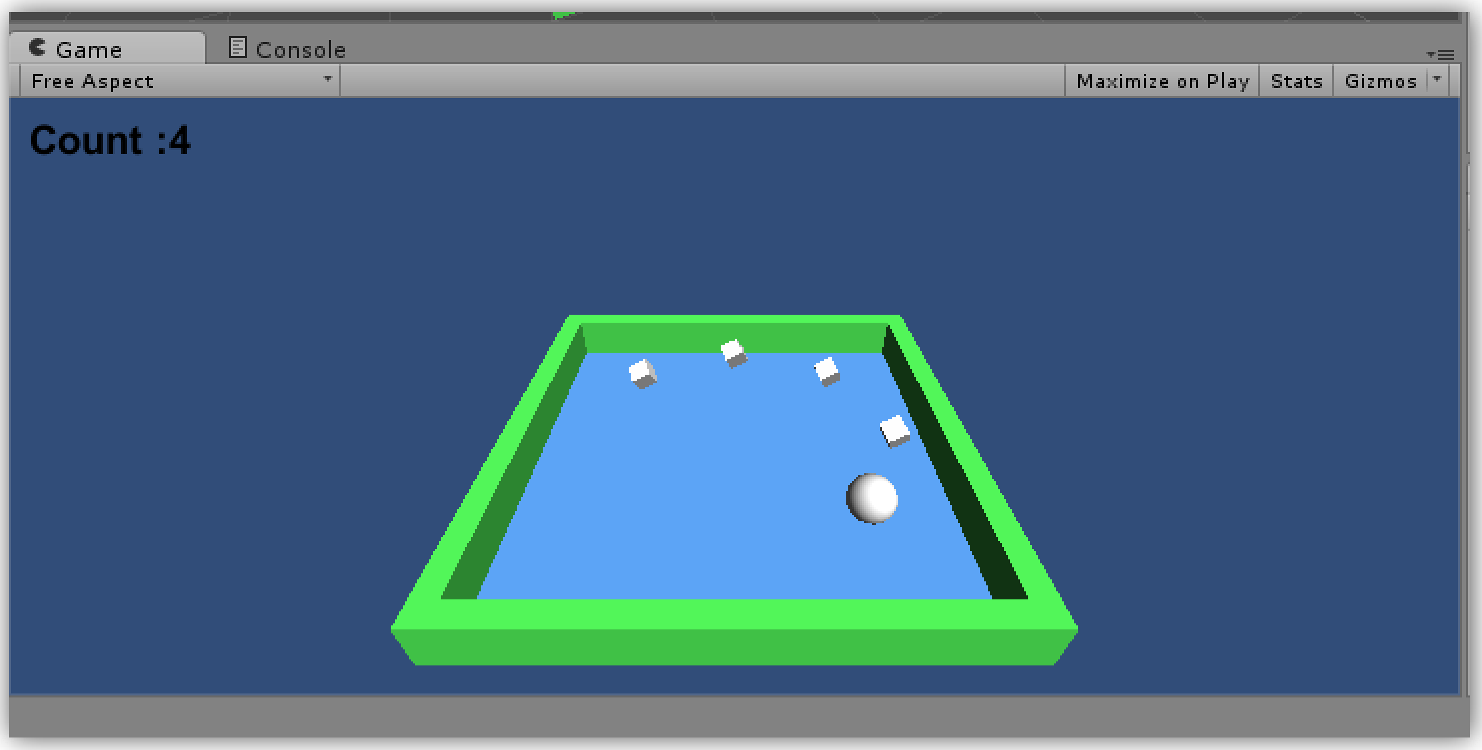
PlayerController و راح نعرف قيم و راح نجعلها تزيد في GUI Text ... لاحظ الكود :

RotateBackups.cs | PlayerController.cs

PlayerController ▸ No selection

```
2  using System.Collections;
3
4  public class PlayerController : MonoBehaviour {
5
6      public float Speed = 200;
7      public float Count;
8      public GUIText myGUIText;
9
10     void Start()
11     {
12         Count = Count;
13         SetCountText ();
14     }
15
16     void Update () {
17         float MoveH = Input.GetAxis ("Horizontal");
18         float MoveV = Input.GetAxis ("Vertical");
19         Vector3 movement = new Vector3 (MoveH, 0, MoveV);
20         rigidbody.AddForce (movement * Speed * Time.deltaTime);
21     }
22
23     void OnTriggerEnter(Collider Col)
24     {
25         if(Col.gameObject.name == "BackUp")
26         {
27             Col.gameObject.SetActive (false);
28             Count += 1;
29             SetCountText ();
30         }
31     }
32
33     void SetCountText()
34     {
35         myGUIText.text = "Count :" + Count.ToString ();
36     }
37 }
38
```

راح نبدأ بالدوال ، لاحظ في الملف اننا عرفت دالتين الاول في Start في السطر ١٠ وهي دالة Static يعني دالة ثابتة ما راح تستدعيها راح تعمل اول ما تشتغل اللعبة ، و في السطر ٣٣ عرفت دالة ثانية بأسم SetCountText و الدالة دي مخزن فيها الـ GUI Text المتغير الي عملته في السطر ٨ ، في السطر ٧ عملت متغير float علشان اخزن القيمة الي راح تظهر في GUI Text لاحظ في السطر ١٢ في البداية راح تكون قيمة المتغير Count هي نفس قيمتة يعني صفر و في السطر ١٣ علشان يطبع لنا القيمة في GUI Text لازم تنادي الدالة تحت المتغير مباشرة ، اما في السطر ٢٨ لاحظ اننا جعلت الكرة كلما تصطدم في Backup راح تزيد قيمة Count بواحد و نفس الشيئ ننادي الدالة علشان تنضاف في GUI Text ، الى هنا ارجع الى البرنامج و راح يظهر لك في الملف خانة فاضية قم بسحب الـ GUI text اليها ، و قم بتشغيل اللعبة :



الان راح نعمل شيئ آخر لكن هذه المرة راح يكون في الكميرا ، راح نجعل الكميرا تتحرك مع الكرة ، قد يكون في نظرك اننا راح نسحب الكميرا الى داخل الكرة بشكل مباشر ، لكن لاحظ هنا ، ان قمت بسحب الكميرا الى داخل الكرة مباشرة راح تتحرك الكميرا مع الكرة باستخدام الـ Position و Rotation ، لكن انا اريد الكميرا تتبع الكرة بحيث ان قيم الـ Position تتغير فقط !! هنا راح نعمل ملف جديد للكميرا راح يكون بأسم CameraMovement الان لاحظ الكود التالي :

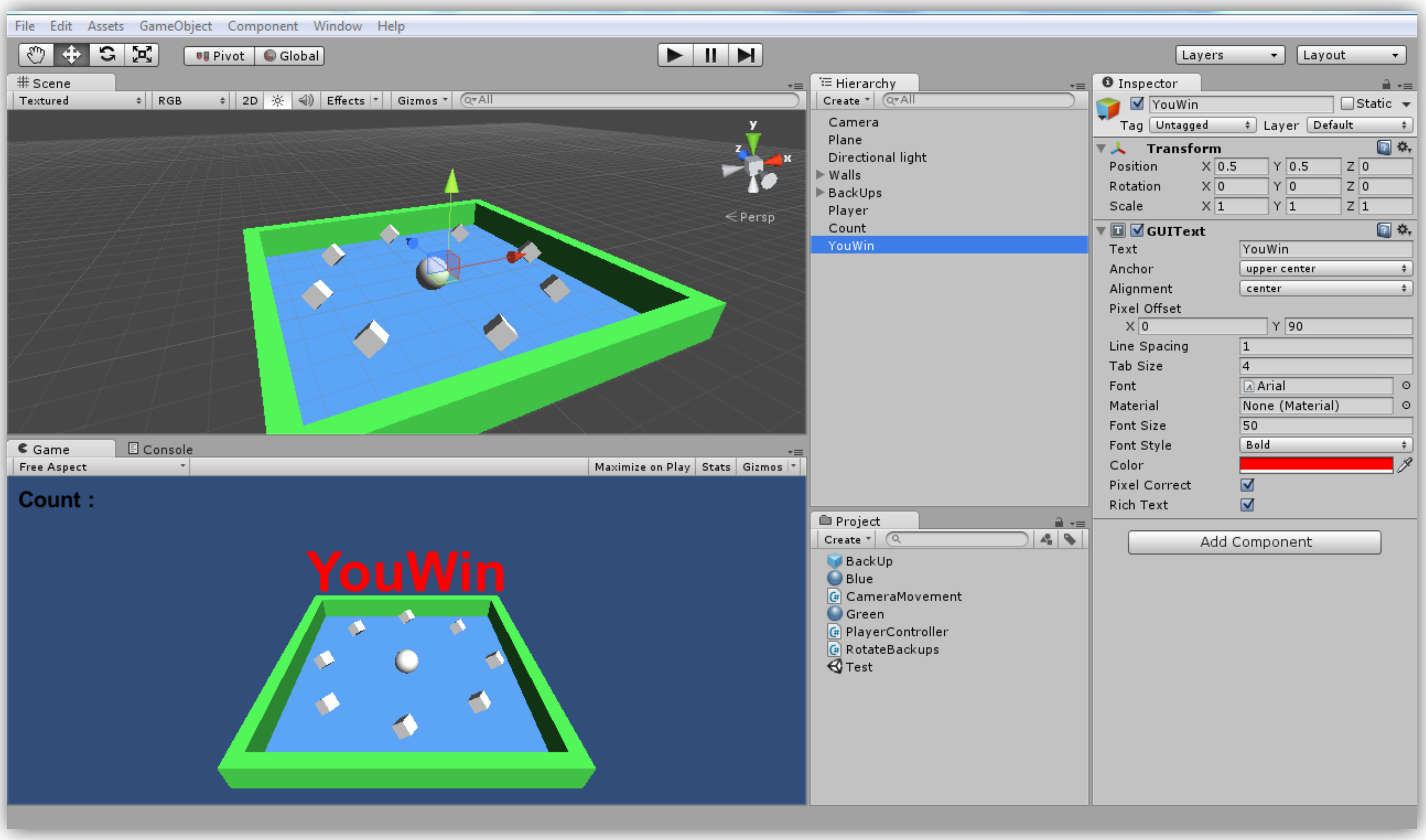
RotateBackups.cs | PlayerController.cs | CameraMovement.cs

CameraMovement ▸ Update ()

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class CameraMovement : MonoBehaviour {
5
6      public Transform Player;
7      private Vector3 offset;
8
9      void Start () {
10
11         offset = transform.position;
12     }
13
14     void Update () {
15
16         transform.position = Player.transform.position + offset;
17     }
18 }
19
```

هذي هي الاكواد الي راح تعمل لنا حركة للكميرا مع الكرة ، في السطر ٦ عرفت متغير Transform و هذا المتغير الي راح نطرح فية الكرة ، في السطر ٧ تحته مباشرةً عرفت متغير من نوع Vector3 و دا الي تتخزن فية المحاور الثلاثة للـ Position و Rotation يعني لازم نحدد نوع الحركة علشان يعرف المتغير اننا خصينا حركة معينة ، طيب في السطر ١٦ انا جعلت حركة الكميرا راح تكون مع حركة الكرة الي هي Player.transform.position بس هنا هو بيطلب مننا متغير من نوع Vector3 علشان تتخزن فية الحركة على المحاور ، هنا انا استدعيت المتغير تبعي offset ،الان ارجع الى البرنامج و اسحب الملف الى الكميرا و راح تلاحظ انة في خانة فاضية هنا قم بسحب الـPlayer اليها ، و قم بتشغيل اللعبة و راح تلاحظ ان الكميرا تتحرك مع الكرة بشكل سلس و ان قيم الـ Position هي فقط التي تتغير ...

الان آخر شيئ علشان نكمل العمل على اللعبة ، راح نعمل GUI Text آخر بحيث ان الكرة ادا لقطت جميع الـ Backups راح تظهر كلمة YouWin و هكذا نكون قدا انهينا العمل على اللعبة ، الان راح نعمل GUI Text قم بضبط موضعه في الشاشة كما هو موضح :



بعد ما تضبط موضعة راح نرجع الى الملف تبعنا PlayerController و الان لاحظ الملف :

RotateBackups.cs | PlayerController.cs | CameraMov

PlayerController ▸ Start ()

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class PlayerController : MonoBehaviour {
5
6      public float Speed = 200;
7      public float Count;
8
9      public GUIText YouWin;
10     public GUIText myGUIText;
11
12     void Start()
13     {
14         Count = Count;
15         SetCountText ();
16         YouWin.gameObject.SetActive(false);
17     }
18
19     void Update () {
20
21         float MoveH = Input.GetAxis ("Horizontal");
22         float MoveV = Input.GetAxis ("Vertical");
23         Vector3 movement = new Vector3 (MoveH, 0, MoveV);
24         rigidbody.AddForce (movement * Speed * Time.deltaTime);
25
26         if(Count == 8)
27         {
28             YouWin.gameObject.SetActive(true);
29         }
30     }
31
```

لاحظ في السطر ٦ اننا عرفت متغير من نوع GUI Text علشان اخزن فية الكائن GUI Text و في السطر ١٦ في Start انا عاملت المتغير YouWin على اساس انة gameObject و بالنسبة لتفعيلة جعلتة غير مفعل بحيث انة في البداية لا يظهر ، اما في السطر ٢٦ عملت الشرط تبعي انة في حالة كان العدد في المتغير Count يساوي ٨ راح يظهر المتغير YouWin او بالأصح راح يتفعل و راح يظهر في الاخير ...

الى هنا و نكون قد انهينا العمل على لعبة Roll A Ball بس هذا ما يعني ان نهاية اللعبة راح تكون هنا ،لا انت الان قم بتطويرها قم بتكبير البيئة او قم بإضافة تفاصيل جديدة او برمجة جديدة ،حتى انك حاول تعدل على الاكود علشان تلقى حلول للأسئلة في رأسك ، و الى هنا ان شاء الله تكونوا قد استفدتم من الدرس البسيط دا ،و الان نستكمل شرحنا عن باقي الدوال و المتغيرات و نأخذ فكرة جديدة منها ☺.

و الى هنا تكون قد انهيت هذا الكتاب و فهمت العديد من المتغيرات و الدوال وعرفت كيف تستخدمها ، المطلوب منك بعد قرائة هذا الكتاب هو فهم كيف تتعامل مع المتغيرات و الدوال و كيف تعمل بعض الالعاب الصغيرة من هذه المتغيرات و الدوال طبعاً هذا الكتاب لا يعني ان نهاية الدورة ستكون هنا لا بل في المستقبل القريب ان شاء الله راح اعمل العديد من الكتب تشرح اشياء مهمة في البرمجة و تساعد في انك تعمل ملفات خاصة باشياء معينة و تسهل عليك استخدامها ، الى هنا ان شاء الله تكونوا قد فهمتهم الكثير من هذا الكتب و علشان تستفيد اكثر حاول تتعلم كل جديد ، اكان بأنك تزور موقع الشركة و تتعلم او اي دروس جديدة ، ونصيحة مني حاول انك ماتتعود على نقل الاكواد بشكل مباشر يعني كل كود تستخدمة في البرمجة لازم تكون فاهمه و في النهاية علشان تعرف ان له وضيفة معينة انت فاهمه مش ناقلها من مكان ثاني و في النهاية كل الذي اطلبة هو دعائكم لي و لوالدي و لجميع المسلمين ، وصلى الله وسلم على سيدنا محمد و على آله وصحبة اجمعين .

Home OF Games Studio